DESIGNING THE SAKAI OPEN ACADEMIC ENVIRONMENT: A DISTRIBUTED COGNITION ACCOUNT OF THE DESIGN OF A LARGE SCALE SOFTWARE SYSTEM

VOLUME 1

A Dissertation
Presented to
The Academic Faculty

By

Klara Benda

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Human-Centered Computing in the
College of Computing, School of Interactive Computing

Georgia Institute of Technology

August 2014

DESIGNING THE SAKAI OPEN ACADEMIC ENVIRONMENT:

A DISTRIBUTED COGNITION ACCOUNT OF THE DESIGN OF A LARGE SCALE SOFTWARE SYSTEM

Approved by:

Dr. Nancy J. Nersessian, Advisor

School of Interactive Computing

Georgia Institute of Technology

Dr. Elizabeth Mynatt

School of Interactive Computing

Georgia Institute of Technology

Dr. Colin Potts

School of Interactive Computing

Georgia Institute of Technology

Dr. Hanne Andersen

Department of Physics and Astronomy

- Centre for Science Studies

Aarhus University

Dr. Geoffrey C. Bowker

Department of Informatics

University of Irvine

Date Approved: [May 12, 2014]

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Nancy Nersessian, for taking me on as a last student, and for the intellectual support and encouragement that she expended for my dissertation work. I also want to thank the School of Interactive Computing at Georgia Tech for the financial support I received throughout my doctoral studies.

Finally, I will also thank my husband, Bence, for embarking on an international adventure with me, and having the patience and good humor to support me in all adversities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Social accounts of technological change make the flexibility and openness of interpretations the starting point of an argument against technological determinism. They suggest that technological change unfolds in the semantic domain, but they focus on the *social* processes around the interpretations of new technologies, and do not address the *conceptual* processes of change in interpretations. The dissertation presents an empirically grounded case study of the design process of an open-source online software platform based on the framework of distributed cognition to argue that the cognitive perspective is needed for understanding innovation in software, because it allows us to describe the reflexive and expansive contribution of conceptual processes to new software and the significance of professional epistemic practices in framing the direction of innovation. The framework of distributed cognition brings the social and cognitive perspectives together on account of its understanding of conceptual processes as distributed over time, among people, and between humans and artifacts. The dissertation argues that an evolving open-source software landscape became translated into the open-ended local design space of a new software project in a process of infrastructural implosion, and the design space prompted participants to outline and pursue epistemic strategies of sense-making and learning about the contexts of use. The result was a process of conceptual modeling, which resulted in a conceptually novel user interface. Prototyping professional practices of user-centered design lent directionality to this conceptual process in terms of a focus on individual activities with the user interface. Social approaches to software design under the broad umbrella of human-centered computing have been seeking to inform the design on the basis of empirical contributions

about a social context. The analysis has shown that empirical engagement with the contexts of use followed from conceptual modeling, and concern about real world contexts was aligned with the user-centered direction that design was taking. I also point out a social-technical gap in the design process in connection with the repeated performance challenges that the platform was facing, and describe the possibility of a social-technical imagination.

CHAPTER 1. INTRODUCTION

Social accounts of technological change make the flexibility and openness of interpretations the starting point of an argument against technological determinism, showing that artifacts are shaped by a rich web of conflicting meanings. The Social Construction of Technologies (SCOT) and the Actor-Network Theory (ANT) approaches insist that technological change plays out in the semantic domain, but they focus on the *social* processes around the interpretations of new technologies, and do not address the *conceptual* processes of change in interpretations. Their accounts describe social dynamics over a static conceptualization of meanings. In my dissertation I will present an empirically grounded case study of the design process of an open-source online software platform, the Sakai Open Academic Environment, to argue that the cognitive perspective is useful for understanding innovation in software, because it allows us to describe the reflexive, possibilizing contribution of conceptual processes to new software and the significance of professional epistemic practices in framing the direction of innovation.

Accounts of conceptual change and design creativity highlight the process of conceptual bootstrapping, whereby novel understandings emerge from existing and familiar conceptual structures. Conceptual novelty is made possible by the generative nature of human cognition, which lends an open character to conceptual expansion. Recent research has further argued that conceptual innovation can be a reflexive process, in which humans create the conditions for their own creativity. In this light, design may be understood as a reflexive epistemic practice, which creates the conditions for its own epistemic growth. In doing this, it also effectuates epistemic selections, which provide directionality to the generative processes of conceptual innovation. The framework of

distributed cognition makes way for weaving the social and cognitive perspectives together on account of its understanding of conceptual processes as distributed over time, among people, and between humans and artifacts.

The case study will show how an evolving open-source software landscape became translated into the vision of a new, web 2.0-based platform, and the open-ended local design space of a new software project, which prompted participants to outline and pursue epistemic strategies for sense-making and learning about the contexts of use. I will describe how a conceptually novel user interface was created in a process of conceptual modeling, and argue that prototyping professional practices of user-centered design lent directionality to this conceptual process in terms of individual activities with the user interface. I will also show that empirical engagement with the concepts of use emerged as a correlate of conceptual modeling, and concern about real world contexts was aligned with the user-centered direction that design was taking.

Social approaches to software design under the broad umbrella of Human-Centered Computing have been seeking to inform the design of new software from a social perspective. In doing this, they have emphasized the empirical contributions of the field and they have overlooked the processes of conceptual construction characteristic of design.

I will point out a social-technical gap in the design process in connection with the repeated performance challenges that the platform was facing. I will describe the possibility of socially informed design in light of the characteristics of the distributed design process of the platform. I will argue that generative conceptual processes are embedded in an open-ended design space, where conceptual processes of sense-making

rely on understandings from the social contexts of use. I will also argue for the importance of framing devices for supporting the social-technical imagination.

I will argue in conclusion that a distributed cognitive outlook to the design of software can serve as a foundation of an account of agency which is attentive to how humans shape technologies through their generative conceptual capabilities, and how they create the conditions of their own epistemic work. With respect to the latter, a distributed account of cognition helps us understand how social and material configurations over human cognition lend directionality to innovation in software, and more broadly within technology.

Outline of chapters

Chapter 1 Introduction

Chapter 2 Problem formulation

The chapter discusses the shortcomings of social approaches in accounting for epistemic processes in technological change, and describes how the distributed cognitive framework allows the integration of social and cognitive approaches in a procedural account of innovation.

Chapter 3 Method

The chapter discusses the qualitative and cognitive-historical method used in the data collection and analysis, and presents the theoretical framework of distributed cognitive-epistemic practices that serves as the theoretical foundation for the work.

Chapter 4 Creating the space for innovation in domain-driven open source

The chapter presents an account of the institutional open-source community of Sakai and its reflexive epistemic practices in making space for domain-driven innovation in open source. The case study describes how the collaboratively outlined vision of a new, web 2.0-based platform gave rise to an open-ended design space, prompting participants to engage in sense-making and learning.

Chapter 5 Constructing a new conceptual model of user experience

The chapter presents a case study of conceptual construction in design, which describes how a novel user interface was created through conceptual modeling in an effort to make sense of the new Sakai.

Chapter 6 UX-driven design

The chapter looks at how professional practices lend directionality to innovation. The case study describes how user interface prototypes frame conceptual modeling in terms of the user interface.

Chapter 7 The social-technical gap

In this chapter I argue that the performance-based challenges of Sakai were indicative of a failure to imagine Sakai's back-end as a socio-technical system; based on an epistemic account of the social-technical gap I am describing the possibility of a social-technical imagination.

Chapter 8 Knowing the contexts of use

The chapter presents a case study on the empirical foundations of design, where I show that knowing the contexts of use is a distributed cognitive process with social sharing and

memory serving as mediators of first-hand knowledge. I also argue for a shift of focus from the empirical to the conceptual for social approaches to innovation and design.

Chapter 9 Infrastructural implosion

The chapter presents a case study of implosive infrastructures, or how component-based reuse of networked software infrastructures contributes to the formation of design spaces, lending momentum to socio-technical innovation.

Chapter 10 Discussion

Presents a DCog account of innovation in software emerging from the case studies, and an outline for social-technical imagination and its contribution to software design.

CHAPTER 2. PROBLEM STATEMENT

Social accounts of technological change have insisted that technology does not unfold according to some deterministic internal logic, artifacts are instead shaped by a rich web of conflicting meanings. These accounts have further suggested that the politics of technological change play out in the semantic domain, through the process of enlisting social support around the interpretation of new technologies. In my problem formulation, I will show that social accounts rely on the theoretical construction of the social distribution of mental contents to account for the process of technological change. At the same time, they disregard the innovative aspects of the creative process underlying new technology and the growing significance of professional knowledge.

On the other hand, social approaches under the broad umbrella of human-centered computing have applied an interpretative framework to the design of interactive software, seeking to uncover the local or broader meanings that people in various contexts attribute to software technology, and to make these available for the purposes of the design of new software. At the same time, considerable discontent has been expressed within the field with respect to the avenues of theory-driven, analytical contributions, and the related possibility of influencing design from a social theoretical standpoint.

Relying on recent work on the generative character of cognitive processes in conceptual change and design I will suggest that the politics of new technology should consider the social distribution of contributions to a cognitive process distributed across people and artifacts. I will argue that an approach based on distributed cognition allows us to engage with the expansive aspects of innovation in technology and to conceptualize the influence of existing technologies as mediated by professional knowledge. In

conclusion to this chapter, I will outline a theoretical framework which combines distributed cognition and social theories of practice.

2.1. Social approaches to technological change and software design

2.1.1. Responses to technological determinism

In the last two decades of the 20th century, the historical-sociological studies of technology and social informatics approaches emerged as responses to concrete forms of technological determinism, in academic discourses with a historical disposition (for the former) and managerial or policy orientation (for the latter). They have sought to reframe the politics of technology by showing that technologies do not follow a technical imperative, they are constructed in social processes, and it is by understanding and engaging in these processes that we may make a difference in technological change. Historical-sociological studies have looked at both past and contemporary technologies, and social informatics has been interested in information technologies in today's context. Both of them have defined themselves as loose research programs, growing out of and fostering a heterogeneous pool of contributors, whose specific stances on the problems of the autonomy of technology and its social impact show some variation, but converge around a few central claims that I will summarize in the introduction.

Historical-sociological studies of technology

The field of historical-sociological studies of technology has been described as integration of the (mainly American) field of history of technology and the (mainly European) sociology of scientific knowledge. It emerged as a productive research program around workshops and in collective volumes, and came to be described as the

Social Construction of Technologies (SCOT) approach. Meanwhile, Bijker's overview (1993) describes the program as consisting of three distinct approaches: the systems approach, the actor-network approach (ANT[a]), and the social-constructivist approach. While the name of Social Construction of Technologies may be confusing, because it references one of the contributing threads, in the following I will use SCOT to refer to the overarching approach, in accordance with widespread usage. Beyond the three threads, the field has also received contributions from the sociology of expectations (Borup, Brown, Konrad, Van Lente, 2006) and the Social Shaping of Technologies (SST) (Williams & Edge 1996) approaches, which I will also reference in my discussion below.

SCOT has emerged as a response to a form of technological determinism based on the assumption of autonomous technology. On this view, technological development unfolds according to its internal technical logic, and impacts society unilaterally (Wyatt 2008). It has also been described as the Schumpeterian linear model of technological change (Schumpeter 1939), which assumes development in distinct, consecutive phases with their own internal logic: techno-scientific invention, economic innovation and socio-economic diffusion. On this account, inventions arise following the trajectory of the growth of science, and innovation translates these inventions into socially useful form, but successful working technologies can fail in the market. SCOT set out to show that throughout their evolution, technologies are shaped by social interpretations of various social actors beyond technologists and managers. In parallel with Bloor's program of the Sociology of Scientific Knowledge, which suggests that nature should not be seen as the cause but the result of scientific beliefs, SCOT has argued that the laws of nature do not

[a] ANT may itself be considered as a loose confederation of contributors.

dictate the success of an artifact, its working is a social phenomenon which requires explanation (Bijker & Pinch 1987, Bijker 1993). People may attach radically different meanings and use values to an artifact depending on their circumstances, meaning that an artifact is characterized by a set of competing interpretations, which underline its interpretative flexibility. The evolution of the artifact unfolds as a tug-of-war between the competing interpretations, each pulling the development in a direction that is favorable to them. This results in a multidirectional process that is subject to social contingency. From a SCOT perspective, what needs to be explained eventually is not flexibility, but the closure of interpretations: why particular interpretations prevail and succeed in stabilizing the form of the artifact. Many of the SCOT case studies set out to describe how competing interpretations converged on a stabilized artifact (see for example Bijker 1997, Law & Callon 1992).

Social informatics

Social informatics (SI) as a research agenda came to be described toward the end of the 20th century to provide a focus to several decades of social and behavioral research on computerization, integrating contributions from information systems research and the social sciences. It has been described as "the interdisciplinary study of the design, uses and consequences of information and communication technologies that takes into account their interaction with institutional and cultural contexts" (Kling 2007). Related research started in the 1970s, when the social context of computerization was limited to organizations, where management attempted to rely on computers to effect change in the operation of the organization. Thus, an important branch of early research focused on the organizational context, which was apparent in theoretical formulations and in the stance

of intervention. Contributors did not only analyze and critique, they also made recommendations for design, or even participated in the design of software systems. With the rise of the Internet and personal computers, SI has come to include a variety of settings, such as education and everyday life. The program of Social Informatics was eventually formulated as a loose agenda, which was applicable to a wide range of research situated within different academic and disciplinary contexts. The Social Shaping of Technology (SST) approach represented a set of parallel aspirations in the UK, with strong connections to Bloor's social constructivist approach in the Sociology of Scientific Knowledge.

SI set out to counter an active form of technological determinism which promoted the development and use of information technologies on the grounds that they were able to impact our lives for the better on the individual, organizational or social levels. In the early days of computerization in organizations, technological determinism appeared as a professional and managerial approach, which sought to achieve better productivity solely through the deployment of technology (Kling 2007). Many SI studies focused on identifying and analyzing the related discourse, as well as disproving the concrete managerial claims about the impacts of IT through empirical research (Kling 1996, Kling & Iacono, 1988, Kling & Iacono 2001, Elliott & Kraemer 2012a). Another focus emerged in response to the claim that technologies prescribe their use, and empirical research set out to show that the meanings and uses of technologies depended on the social context (Kling 1991). Related contributions include Kling and Scacchi's work on the web of computing (1982), Star's ethnography of infrastructure (Star and Ruhleder 1996, Star 1999), and Orlikowski's adaptation of structuration theory (1992, 2000). Empirical

approaches based on interviews or ethnographic observation were seen as providing richer and more accurate understandings of a complex socio-technical reality than the simplifications offered by technological determinist discourses, and hence better positioned to inform the design of information systems. Recommendations from SI have focused on human-centered systems (Kling and Star 1998) and user interaction with technology. Thus, recommendations from the study of the adoption of an office communication suite emphasized the differences in local contexts, and the related need for more flexibility in the software, so that users could customize and adapt the tool to their own work setting (Orlikowski, 1993).

2.1.2. Two threads in social accounts of technological change

Social accounts of technological change brought together two threads of theoretical conceptualizations:

(1) One centers on the configuration of actors around the evolving technologies.

(2) The other centers on the meanings that envelope and shape technologies.

Related to the first thread, SCOT talks about relevant social groups (Pinch & Bijker 1987, Bijker 1997), who are constituted through the interpretative stance they take toward a new technology. While relevant social groups may arise from existing social groupings, such as users, producers, or various roles within organizational contexts, existing social groups become redefined through their interpretative relationship to the artifact, and may be fragmented into sub-groups, as described in the case of the danger-seeking youths and the bicycle (Bijker 1997). Callon has produced analytical concepts around the principle of generalized symmetry (Callon 1986b, Callon & Latour 1992), which holds that artifacts and social actors co-constitute each other. He has suggested that

the development of technology calls forth the creation of an actor-network, which gains stability from relating various actors and evolving artifacts to each other through a reciprocal translation (Callon 1986b, 1987): in the case of the electric vehicle, Callon (1986a) described how the French electricity company EDF created a master scenario around the future development of lead car batteries framed by the social movement toward more democratic and less polluting transportation. This scenario effectuated a series of translations: users were translated into socially disgruntled agents, who sought a particular version of social reform, the lead batteries were translated into good, environmentally friendly car batteries, and Renault into a company considering a radical redesign of the car. The translations involve a simplification of the different agents according to the logic of their juxtaposition with other elements (a process elsewhere described as punctualization elsewhere (Law & Callon 1992)). Stability of development projects and technologies comes from the success of an agent in tying these various actors into the knot of an actor-network. Actor-networks could however come loose if the agents started to resist the translation. In the VEL case described by Calln, the lead batteries proved to be inefficient and polluting, Renault saw a different opportunity around the car of the future, oil prices went down, cars became more environmentally friendly and cheaper, and users were after all not so socially disgruntled.

In discussing SCOT and SI alongside each other, we have to be aware that they represent distinct styles of theorizing. SCOT has been characterized by a reflexive stance that stands outside of the process studied, which results in a panoptic view and conceptualization of the phenomena (Bourdieu 1977), while SI has been speaking from a perspective that seeks to be part of the change. Thus, SI has had no overarching theory

similar to relevant social groups, but studies have typically devoted considerable attention to making sense of the social context of technology use, and the various groups and perspectives which participated locally. In a similar vein, SST approaches have talked about the social configuration of social constituents (Williams & Edge 1996). The distinction has been commonly made between users, designers (or technologists) and managers (see for example Kling and Star 1997), who could be further differentiated into subgroups according to the local contexts. Related to local groupings, Star and Ruhleder (1996) talked about how infrastructures were locally embedded in the social context and learned as part of social membership.

The second thread in the social analysis of technologies concerns the realm of understandings through which humans relate to technologies. The notion of social construction references the centrality of human interpretation in shaping the form of artifacts. SI has also relied on a range of mental concepts, talking about the importance of user understandings and user education (Sawyer and Rosenbaum, 2000), or the role of visions and discourse in the mobilization efforts of computerization movements (CMs, Kling & Iacono 1988, Elliott & Kraemer 2012a).

Both SCOT and SI have viewed interpretations as social insofar as they are projected onto a grid of social space. Social studies of technological change have used the concept of frame to capture the distribution of mental contents in a social space, and theorize the connection between interpretations and social groups, albeit with some differences of focus. Bijker (1987) has defined technological frame as composed of "current theories, goals, problem-solving strategies, and practices of use", which underpinned the meanings attributed to an artifact. Technological frames are rooted in the

social context, and are thus linked to unfolding interactions between social groups. Frames defined in this way acted as a conceptual "hinge", referencing the mutual constitution of social groups and interpretations amidst shifting social situations (Bijker 1992), as theorized by Callon's notion of generalized symmetry (1986a).

Within SI, Orlikowski & Gash (1994) borrowed the concept of frame from cognitive psychology to describe how the values, views and overall perspective of different social groups shape their understanding of new technologies. Their definition of technological frames includes assumptions, expectations and knowledge about the technology, and specifically its conditions, applications and consequences in particular social contexts. Finally, studies of computerization movements have relied on Goffman's concept of frames (1986) as used in the sociology of social movements (Snow & Benford 1992). Iacono and King (2001) have further suggested that they drew on both Bijker's (1997) and Orlikowski and Gash's (1994) previous work. Technological action frames were described as interpretive schemas shared within larger social groups, which focalize ("punctuated") meanings attributed to new technologies, and thus make discussion about them possible. They describe how a technology works and what future would be like with the technology (the framing aspect). They legitimate adoption of the technology by providing high-level guidelines for using it in the immediate local context, but did not engage with the details of usage. The guidelines provide a value-driven line of action based on the diagnosis of present problems and a prognosis of better outcomes (the action aspect). Studies have since described a wide array of CMs, including office automation and productivity, artificial intelligence, personal computing, the Internet, the semantic web, distributed collaboration, and ubiquitous computing (see the volume edited by

Elliott and Kraemer (2012a)). Like SCOT, SI has also emphasized the changing nature of computerization discourses (Elliott & Kraemer 2012b) and the importance of learning from and by users (Sawyer & Rosenbaum 2000). It may be further suggested that the mobilizing nature of technological frames parallels SCOT's principle of general symmetry between social groups and interpretations: social groups are pulled together around mobilizing discourses.

While the approaches agree that social interpretations are contentious without being openly conflicting (Bijker & Law 1992, Kling & Iacono 1988), as suggested by the terms of interpretative struggle and controversy, their differences in theoretical orientation have lead them to consider conflicting frames of new technologies in radically different light. For SCOT, interpretations are not rooted in an external reality which could lend truth to them, and no argument or claim is intrinsically truer or better than another. SI similarly adheres to a constructivist stance, but unlike SCOT, it has reserved space for critique. Thus, Kling & Iacono (2001) have insisted that computerization discourses should not be scrutinized in terms of their truthfulness to reality, but their selective reliance on meanings in the social context. The SI approach has further sparked a critical line of research on CMs, which has denounced some forms of technological discourse as unfounded and misleading (see the papers in Elliott & Kraemer 2012a).

This divergence over the role of reality has significant consequences for the way technological imagination has been treated in SCOT and SI. SCOT has been receptive of the description of imaginative forms, notably in the sociology of expectations (Borup et al., 2006). The sociology of expectations has theorized the importance of a particular form of future-oriented thinking in technology development, described as imaginings,

expectations, visions or promises. On the basis of related ethnographic and interview research, they have characterized these projections as abstract conceptualizations which were loose enough so that they could change or could become gradually filled in throughout the course of development. They have also been described as images combining technical and social aspects of technologies in a tightly interrelated manner. Researchers of expectations have argued that conceptualizations of future technologies were central to understanding the process and direction of technology development. Because of their abstract and flexible nature, these conceptualizations provided a platform that was essential for the coordination between contributing actors, both at the initiation of a project and in the course of its evolution. Callon's theory (1987) around translation has essentially described a similar process, whereby the narrative construction of an actor-world configured actors within a social space of development by means of simplificatory conceptualizations of their nexus with the evolving technology (translations).

Kling & Iacono's technological action frame (2001) refers to a parallel set of future-oriented, visionary forms in public discourse, which create mobilization behind CMs for innovation in computer technology (Kling & Iacono, 1988). While the sociology of expectations has remained neutral toward its topic, CMs have been denounced by SI research as utopistic, visionary and ideological, notably on account of the unfounded promise of an equally beneficial computing for all, which suggested that social groups were configured in a level space around technologies. In response to CMs, SI has argued that design of information technologies should follow the approach of critical realism,

addressing the empirical complexity of the social world with the theoretically and empirically informed epistemic arsenal of social study.

In sum, the concept of frame has emerged as an attempt to connect cognitive and social levels of analysis. SI in particular has drawn explicitly on previous research about interpretative schemas and frames in social and cognitive psychology. Meanwhile, cognitive contents have been treated as static and given, and the mental processes of meaning-making and construction did not have a role to play in the overall theory. The dynamics of frame theory played out at the level of social configurations as a reshuffling of these contents. Research on cognition, as I will elaborate later on in the chapter, provides insights on the mental level of the same dynamic, suggesting that frame analysis may be taken further in a way that combines social and cognitive *processes* in the account of the evolution of technologies.

2.1.3. Human-centered approaches to the design of software

The application of the social perspective to the design of software has resulted in a third set of social approaches, which include the field of Computer-Supported Cooperative Work (CSCW) as well as socially-oriented contributions within the field of Human-Computer Interaction. Lines of demarcation are often difficult to draw in this domain. Grudin (1991) has for example suggested that the field of CSCW was created at the boundary of two parallel research communities, Management Information Systems research and Human-Computer Interaction, which were exploring the shared problem of software design from within two distinct organizational contexts. Social Informatics had connections to both of these communities.

Attempts to bring these various perspectives together has existed for a long time, and the human-centered label has been repeatedly applied with this intention. Kling and Star (1998) described human-centered information systems from within Social informatics as an interdisciplinary approach to the research and design of software systems characterized by growing complexity:

> "The promise of human-centered systems is that knowledge of human users and the social context in which systems are expected to operate become integrated into the computer science agenda, even at the earliest stages of research and development." (para. 3)

Georgia Tech chose the label of Human-Centered Computing (HCC) for a PhD program which was designed to extend the traditions of Human-Computer Interaction. One of the founders, Jim Foley defined HCC as:

> "the science of designing computations and computational artifacts in support of human endeavors" (cited in Jaimes, Sebe, Gatica-Perez, 2006: 856).

Most recently, Bannon has suggested that a new perspective on has been emerging (2005; 2011) with:

> "a more holistic view of human-systems interaction that begins to privilege the human, social and cultural aspects of computing." (Bannon, 2005: 32)

Elsewhere, he listed Human-Computer Interaction, Computer Supported Cooperative Work, Participative Design, Interaction Design and Social Informatics as "human-centered" fields involved in the human and social side of the computing discipline (Bannon, 2011). I will use this label in the broad sense outlined by these various contributors to refer to the approaches within the various fields that have attempted to

emphasize the social and cultural aspects of human engagement with software technologies.

Human-centered approaches bring a strong empirical contribution to the future-oriented, imaginative practices in design. Ethnography has been a method of choice for a wide range of related research. Advocates of ethnography in human-centered computing have emphasized the importance of understanding the complexity and rich detail of local settings, and the role of the researcher in making available tacit knowledge about the everyday.

At the same time, social researchers have expressed a sense of marginalization with respect to the scenes where software is made, and there has been a related sense of discontent about how ethnographic contributions are being used. Dourish (2006, see also Dourish & Bell, 2012), for example, has suggested that the prevailing practice of using ethnography toward the formulation of requirements and implications for design is limiting, both in terms of the spread and the depth of its contribution. He has advocated for a more general reliance on input from ethnographic research across the design process, and more space for social theory in informing design. Rogers (2006) has pointed out that human-centered computing has an affinity for the high-level framings of computerization movements, such as ubiquitous computing, but cannot live up to its generalizing claims, such as making the environment smart. She suggested that design should be instead attentive to the local context and the goals of users. Finally, Bardzell (2010) has suggested that designers taking an advocacy position run the risk of imposing their own understanding of an improved world, emancipation and progressive design. All of these accounts grapple with the implications of a world-to-be inherent in unfolding

technology, yet human-centered computing has had limited interest in imagination, and in the relationship between empirically grounded and imaginative forms of thought.

More recently, Dourish and Bell (2012) have argued for future-oriented, imaginative approaches in ubiquitous computing, and Bannon (2011) has emphasized the constructional, exploratory and imaginative character of the design of social software. At the same time, these suggestions have remained at the level of insight, and human-centered approaches have attempted no systematic account of the future-oriented, tentative forms of thought in design. Instead of exploring the dynamic mental processes behind the imaginings, expectations, visions and promises, as suggested by the sociology of expectations, theoretical accounts of HCC practice tend to fall back on such static concepts as needs, goals, or values when referencing the domain of users.

2.2. Cognitive processes in design

2.2.1. Research on design and creativity

Research in design has explored the conceptualizations and thought processes underlying the evolution of artifacts. It is a generally accepted starting point in design research that artifacts go through various stages of tentative forms in the process of design (Liikkanen, Laakso, Björklund, 2011; Visser 2006), and designing involves some form of mental activity. There are a wide number of cognitive studies on design, which describe the general outlines of the thought processes involved in designing (see the overviews in Liikkanen et al. and Goldschmidt & Badke-Schaub, 2010), and mental processes appear centrally in research without a cognitive focus. Thus, Schon (1999) describes the reflexive conversation with the situation as an epistemic stance, and for Bucciarelli (1994), the concept of object-world is informed by professional knowledge. Design

research has also shown that design thinking is not confined to the head, and tentative forms may serve as external, material aids (Christensen & Schunn, 2007; Visser 2006). In the simple case documented in countless case studies of design research, the artifact starts out as a brief, usually in the form of a printed document. As designers start working, they create sketches, mockups, prototypes, technical drawings, CAD visualizations and other models to provide form to the artifact. These renderings are fragmentary and tentative: they are true to some features of the artifact, and they allow designers to engage with the artifact from various angles of abstraction. The future-oriented conceptualizations described by the sociology of expectations fit well into the above list, as tentative artifacts in a verbal form.

A central concern in design research pertains to the specificities and regularities of the thought process involved. One line of research has been focusing on establishing the general outlines of the problem-solving process in design. Studies have come to converge on the view that design is characterized by a co-evolution of problem and solution (Dorst & Cross, 2001), rather than a direct and orderly path from a fully specified problem to the solution, as was suggested in early models, notably formulated in engineering management (Pahl, Beitz, Feldhusen, Grote, 1984). Designers cycle through a number of tentative solutions, and as they fill in particular aspects of the new artifact with new details, they also learn more about the problem and the context, which can take the evolving artifact in unexpected directions. At the same time, empirical studies have found considerable differences in the overall problem-solving approach across different professions and organizational contexts (Kruger & Cross, 2006).

Another line of research on design creativity has focused on describing local cognitive processes, and has come to suggest that routine design processes accommodate moments of creative or expansive designing, which result in unforeseen and unexpected solutions that take the process beyond the realm of familiar designs. Related to modeling creativity in artificial intelligence (AI), Gero (1990) formulated a theory of expansive design which attempts to model the cognitive process whereby designers combine contextual knowledge with structured schemas of design. Gero's account is based on a separation of knowledge involved in design into (1) general knowledge about the context, and (2) design-specific professional knowledge, which echoes the widespread systems engineering approach (Pahl et al., 1984). On the one hand, design activity is understood to be oriented toward the creation of artifacts operational in a natural and social world, which imposes *constraints* on possible designs. Thus, design activity depends on the designer's perception of a relevant context, which may change depending on their perception of the evolving design. On the other hand, designers bring *design-related conceptual structures* to bear on their designs. Gero describes knowledge about designs as schemas at different levels of abstraction. Central among these schemas are design prototypes, which bring together function, structure and behavior in one schema. The prototype references a state space of possible designs, which may be defined as the particular instantiations that the schematic structure allows. Routine designs repeat familiar surface structures of designs covered by the schema, while innovative designs rely on the structures in novel ways. Finally, creative designs represent a third category, which goes beyond the state space referenced by the schema by giving rise to novelty that does not follow structurally from the designer's design schemas.

AI theory of design creativity is not interested in the social-organizational context of design. While design research has shown some awareness of this context, both lines of research may be characterized as focusing on cognitive dynamics, and do not include the social distribution of cognition in their explanations. In this, they parallel the one-sidedness of social accounts, which leave cognitive process out of the explanation. To view the social and cognitive levels of analysis in interrelation, we need a framework that provides explanatory access to the social distribution of cognitive process. In the following section, I will discuss the solution distributed cognition brings to this problem, and more specifically how it has been applied to the context of science and engineering design.

2.2.2. Distributed cognition in design

Distributed cognition (DCog) has been formulated by Hutchins as a framework for construing cognitive processes as comprising mental processes, social interactions and material artifacts (Hutchins 1999; Norman 1991; 1993, Hollan, Hutchins, Kirsch, 2000). Hutchins has suggested analyzing the thinking process outlined by cognitive science as spreading beyond the mind of the isolated individual into the world (Hutchins, 1995b). This implies a threefold distribution of cognitive process (1) in time, (2) between humans and artifacts, and (3) in the social space across various actors (Hutchins, 1995a; 1996). When seen in parallel, the three layers of distribution provide analytical access to creative intellectual work in organizational contexts, such as labs and projects, or the wider social timeframe of culturally inherited cognitive activities and artifacts, such as navigation and maps. DCog has become popular in design, and notably in the design of computer-based interfaces which need to take into account the cognitive processes behind existing tools to

be replaced by digital solutions (Nardi & Miller, 1991; Petre & Green, 1993; Hutchins, 1995b; Hollan, Hutchins, Kirsch, 2000; Liu, Nersessian, Stasko, 2008). By their nature, these studies tend to focus on established cognitive processes that are already in place. The DCog framework has also been used in studies of science and design settings to describe how visual and gestural representations and artifacts enter into or coordinate local processes of meaning-making (Becvar, Hollan, Hutchins, 2008; Goodwin, 1995; Perry, 2003; Visser, 2007; Vertesi, 2012; see also the theoretical overview in Kirsch, 2009).

Meanwhile, as Hutchins has also suggested (1996), the framework can be used to study distributed processes of cognitive change, including the social construction of meaning around new artifacts, or how people

> "create their cognitive powers by creating the environments in which they exercise those powers" (Hutchins, 1995a: xvi).

Addressing conceptual change in scientific and engineering research, Nersessian has adapted the DCog framework to account for the generative nature of distributed thinking processes, showing how the expansive character of conceptual change and creative design is entangled with meaning-making in a socially and materially distributed setting, and how participants act on and change their own environment to scaffold epistemic expansion. Her work extends the DCog framework from the study of cognitive processes underpinning existing artifacts to creative cognition in design, and spells out an integrative social, cultural and cognitive theory of creative construction.

Research on conceptual change in science has originated (among others) with Kuhn's (1970) work on normal and revolutionary science (Nersessian, 2002b; Andersen,

Barker, Chen, 2006). The Kuhnian account frames the history of sciences in terms of paradigms, which exhibit conceptual coherence internally, and incommensurability of concepts between themselves. Kuhn's formulation raises the problem of how conceptual novelty may be accounted for with routine scientific activity at the backdrop: "how scientists build on existing structures while creating genuine novelty" (Nersessian, 1992a: 9). Nersessian's account (2008a) describes the development of new concepts as a modeling process, which relies on the tentative formulation of relational structures called models, abstracted from existing structures of representation in problem-solving processes. These models, which are often dynamic, come about through such common cognitive processes as analogical and visual inference or mental simulation, and they are continually adjusted in light of their fit with available representations of phenomena. Creative solutions arise through combining novel constraints in model-building processes. This process has been referred to as bootstrapping (Nersessian, 2008a; Carey 2009), because it involves the creation of representational resources that make conceptual expansion possible, resembling the paradoxical situation of pulling oneself up by one's own bootstraps, or reaching out to new conceptual structures by the conceptual structures one has available. Scientists create the conditions of conceptual change by constructing models that act as cognitive space for exploration through cognitive tinkering.

Research in science and engineering has become increasingly collaborative, and Nersessian has relied on the DCog framework to study how cognitive bootstrapping processes unfold in the collaborative context of the research laboratory (2008, 2012). Her ethnographic studies of engineering design in research labs has also shown that modeling may rely on material aids, such as diagrams, devices, or even complex arrangements of

instrumentation, and these may be passed on or shared across contributors. Her work documented various examples of complex material models, which are implicated in a process of cognition distributed across humans and artifacts. In this context, where the goal is the design of novel technical artifacts, the various models can work both as tentative formulations of the future artifact and representations that are implicated in the cognitive modeling process. The engineering lab is described as a socially and materially distributed problem space and learning system, where researchers at various stages of their career partner with material artifacts produced throughout the collaborative history of the lab to explore possibilities of their expansion.

2.3. Participation in cognitive process

Politics of technology centers on the problem of participation, and more specifically, whose knowledge and understandings inform new technologies (Suchman, 1994). Social accounts of participation focus on the distinction between makers and users, or across different professional groups, and specifically between technologists and others.

In SI, and especially in HCC, the problem has been described as the importance of knowing user needs for better serving them through design, an approach shared by the wider community of human-centered design (Grudin, 1992; Norman, 1988). The underlying conceptual construction has been well captured by Ackerman (2000) in the term socio-technical gap, which has been coined to describe to the discrepancy between a flexible and nuanced social context and technology which is by nature inflexible,

> "the divide between what *we know* we must support socially and what we can support technically" (p. 179).

Many contributions in SI or HCC have followed the related ethnographic agenda of channeling knowledge about or from the users to the process of design to make systems more usable (Norman, 1988), more appropriate to local contexts of use (Suchman, 1995; Star, 1999; Dourish & Bell, 2011), or capable of fulfilling an emancipatory potential (Bardzell, 2010). The major epistemic division lies between users and makers, or more generally between knowledge of the social context and knowledge of technical design.

The concepts of frame and relevant social groups introduce the possibility of further epistemic distinctions, and analyses have described the nature and significance of the contributions from technical professions. Callon (1987) has argued, for example, that engineers act as sociologists in designing social futures for future technologies. He has hinted that they are even better than sociologists, because they can make these futures come true, or at least put them to a trial of strength in the real world. In his study of Aramis, another transportation project, Latour (1996) has also acknowledged the proliferation of social narratives, but contrary to Callon he suggested that engineers did not love their sociology because they were more interested in furthering technical knowledge than changing society, and their social narratives did not play into the expansion of technical knowledge that resulted from their engineering work. Finally, in advocating for implementing the reflexive practice of the social sciences in computing, Agre has suggested that "computing has been constituted as a kind of imperialism; it aims to reinvent virtually every other site of practice in its own image" (1997: para. 2).

In general, social accounts have acknowledged the importance of epistemic perspectives that inform evolving technologies, and researchers have investigated what the preponderance of engineering or designer knowledge might mean for the direction of

technical change. Meanwhile, as I have pointed out before, they have not provided the analytical tools to explore how the different epistemic perspectives contribute to the process of the creative cognitive process in design and the underlying cognitive expansion. My claim is that the DCog framework can provide a powerful analytical tool for making sense of participation and politics based on the cognitive processes involved in the design of new technologies.

To understand participation, we should map the social configuration of contributions to the cognitive process of design, and to the nature of the contributions of different social constituents: what are the forms of knowledge that relevant social groups contribute to design, and most importantly, whose knowledge contributes to the generative forms of thinking that underlie the dynamics of design. The analysis should be attentive to the distribution and combination of diverse contributions throughout the temporal evolution of design, their role in the processes of bootstrapping and expansion, and their engagement with social and technical aspects of socially embedded technologies. We should be wary of a priori distinctions in the nature of the epistemic contribution from different domains of knowledge, such as the user and maker, social and technical, or contextual and design knowledge; in line with the principle of general symmetry, the related distinctions should be expected to be shaped by, rather than simply pre-exist, the social configuration of design. Finally, the importance of tentative material forms has been noted repeatedly; it may be expected that different social constituents bring different resources and skills for engaging with various types of artifacts. Thus, it is important to chart the role of tentative artifacts in connection with the professional groups that rely on them.

2.4. Revisiting context in light of distributed cognition

DCog outlines three dimensions in the distribution of cognition: social, material and historical. Of the three dimensions, social studies have been primarily focusing on the social context: the social distribution of meanings around technological change. In the above, I have shown that in doing this, they relied on a static understanding of cognition as cognitive contents, such as understandings, interpretations or schemas, and I have argued that we should consider the social configuration of the cognitive process instead. I will now turn to the other two dimensions, and argue that they should be equally considered when we are making sense of the context of design. First I will argue that the social configuration around new technologies accommodates large scale, historically situated patterning, with local processes of reconfiguration. Local reconfiguration may involve the reflexive activity of participants to create environments that are more conducive to preferred processes of creativity. Second, I will show that existing technologies serve as context for ongoing design work, and influence technological change through the mediation of professional practices.

While SCOT in general has not been particularly interested in describing large scale patterns in the social configurations of relevant social groups, Hughes brought a historical perspective to the analyses, describing two historically distinct socio-technical webs (Hughes, 1996) of technologies: systems and projects. His early work described the growth of a seamless web of technical networks and large-scale organizations with significant bureaucracies. This line of research has continued under the label of large-scale technologies or infrastructures. More recently, Hughes has described the emergence of the project as a new organizational form in the second half of the 20th century (Hughes,

2011). While large-scale technologies follow a history of slow motion, rolling forward a momentum from the past, projects are quick-paced, future oriented collective design endeavors seeking to generate change in technologies. It may be noted that other case studies in SCOT describe a third pattern, which I will now refer to as the mass market, where technical change plays out in a cyclic process of feedback between producers and users of technology. It seems that these historically situated forms should be seen as ideal types (Weber, 1904/1949), which appear with significant variations, and may even play into each other, as we see when projects are created for the iterative redesign of mass market artifacts, such as in the case of the Sony Walkman (Du Gay, 1997). At the same time, the three processes are considerably different, which means that findings from cases in one social organizational form will not be directly translatable to others. The case studies of the bicycle, the fluorescent light, the bakelite (Bijker, 1992; 1997) or the car (Kline & Pinch, 1996) show cyclic processes of construction in the market, notably involving user feedback. On the other hand, most case studies from the sociology of expectations (Van Lente, 1993; Van Lente & Rip, 1998) and ANT analyze one-off projects, which were based on the gradual crystallization of their own social platforms of collaborative negotiation (Law & Callon, 1992; Callon 1980; 1986a; Latour, 1996).

In contrast to the production system instituted at the turn of the 20th century, and mostly built around Fordist principles of mass production, Hughes (2011) describes projects as interdisciplinary professional communities or joint ventures, which are characterized by loose horizontal, distributed coordination through consensus and meritocracy rather than strong top-down management control, and follow an open-ended construction process which seeks constant novelty rather than incremental change.

Design research appears to be describing this latter organizational form, which makes space for collaboration around the iterative elaboration of tentative projected artifacts.

Studies of how work gets distributed indicate a second, equally important way in which the social and cognitive aspects of the process are interrelated. As I have previously suggested, creativity may be understood as a bootstrapping process, whereby participants create the conditions of their own creativity. Hall, Wieckert and Wright, (2010), for example, have described how linguistic tools were used to make sense of and design future work: in one case study entomologists were seeking to expand their contribution through the adoption of a statistical method borrowed from another field of population biology. They discussed narratives of future work, inserting the method into their local work process in different ways. An initial narrative of adoption was countered by a narrative that challenged the contribution of the method, and eventually, a third narrative emerged, which described a novel use of the borrowed tool to provide beneficial contribution to the ongoing work. In this process, the researchers could be seen redesigning the context of their own epistemic work, and devised a creative solution for this purpose. Nersessian (2012; see also Harmon & Nersessian, 2008) has described situations where instead of borrowing, researchers built their own tools for scaffolding creativity. She has shown how researchers were building complex material instruments to model the biological process of blood flow, and how computer modeling was being used to grasp possible approaches to patterning in a neural network. Beyond material construction, these reflexive activities could also involve social arrangements: in the neural lab, several researchers entered into collaboration on the basis of the computational tool for modeling the neural network. In light of the above, projects may

be seen as working on the creation of a protected space of work, where participants are encouraged to shape their local work environment by designing the way work gets done, and thus creating the conditions for their own creativity.

Projects also bring together social constituents with different perspectives and epistemic resources for engaging with the evolving artifact, which necessitates an account of how different perspectives and approaches are combined. Social studies have conceptualized this process in semantic terms, either as a verbal process of negotiation which reaches closure through consensus, or as a successful process of translation accomplished by a central actor (an obligatory point of passage), who is able to define the role of other actors and material artifacts (Callon, 1986a). Studies of design suggest that beyond semantic processes, artifacts also play a mediating role in design. In these accounts, different professional groups engage with tentative objects from the perspective of their frames and relying on their epistemic resources (see Bucciarelli's (1994) related notion of object worlds), and they interface through the artifacts acting as boundary objects (Star & Griesemer, 1989; Henderson, 1991), rather than on the basis of mutual verbal understanding or agreement. The result of such processes has been described as a heterogeneous joint consensus, which does not rely on full mutual understanding and acquiescence of the collaborating partners (Andersen, 2010).

Finally, artifacts are becoming increasingly systemic, referencing the wider technical context at the particular historical moment of their design. They are not only built with available parts of varying complexity, they also rely on conceptual schemes of existing artifacts. Thus, design research has argued that new designs often follow the prototypes of existing artifacts, which act as exemplars for approaching design problems.

In social studies of technology, paradigm is a related concept, which has been used to argue for the existence of technical trajectory (Dosi, 1982). Social studies of software have noted the emergence of information infrastructures, which have been described as stacks of various layers of software functionality. Information infrastructures accommodate and invite openness through extension, acting as the driver in a process of unbounded evolution of supported software technologies (Hanseth & Lyytinen, 2010). The various technical forms may be imported into a project in a variety of forms, ranging from abstract models to concrete technology solutions, but in all instances they become coupled with the conceptual resources of the actors who engage with them. Thus, it may be said that the external technical context becomes mediated by different forms of locally available expertise.

2.5. The practice framework

In the above I have argued that addressing participation in technological change as raised by the social accounts requires that we rely on a socio-cognitive approach in the analysis of social construction through interpretations. At the same time, the approach needs to be responsive to the centrality of material artifacts and cultural-historical arrangements implicated in the process of development. I will now describe a theoretical framework that captures the cognitive, historical, social and materially grounded nature of the shaping of technologies, and serves as a theoretical lens to guide the research.

Tweney (1989) suggests that to guide research we use interpretive schemes that are different from theories obtained in research. These structured understandings are informed by theoretical abstraction, but they are different from theories. Tweney calls these schemes ‘frameworks’, and I will adopt this term.

The proposed research relies on a framework that places DCog within the broader perspective of a practice-based approach. Both of these have been established as valid frameworks of social research, but the online setting of the case will require an application of the notion of cognitive practices to a context mediated by networked communication technologies. DCog and the practice approach have distinct origins in the cognitive and social sciences, but they converge around the understanding that their central phenomena are emergent from human activity anchored in a material world.

I have discussed DCog at length in previous sections, showing how it has overturned the central assumption of cognitive science that cognitive processes should be located within the head of a single individual. For his initial formulations of the framework, Hutchins (1995a; 1996) relied on extensive ethnographic observation of navigation activities on board of a ship to show that the cognitive outcomes of wayfinding emerge from a process comprising persons and artifacts. While Hutchins made space for the study of both established and creative processes of cognition, subsequent applications by himself and other researchers tended to focus on cognitive processes with established procedures and artifacts, and how these come to be used in the context of novel problems (e.g. bringing the disabled ship into port). Nersessian has further developed the framework through developing an integrative socio-cognitive account of processes of the creation of novel procedures and artifacts.

The practice approach has a heterogeneous origin within the loose domain of social theory. In the second half of the 20th century, several authors formulated theories that centered around some notion of practices, among them Ludwig Wittgenstein (1953), Pierre Bourdieu (1977), Anthony Giddens (1979; 1984) or Michel Foucault (1976; 1980).

While the shape and thrust of these theories show significant differences, they share in the attempt to go beyond earlier theoretical frameworks which posit social structures or individual action as the building-blocks of social phenomena. Practice is introduced as a bridging concept between the individual and the social. In his overview of practice approaches, Schatzki (2000) captures the commonality across different formulations by describing practices as arrays of human activity, which rely on a convergent set of understandings or skills, and are the locus where the persistence and transformation of social life can be grasped. Practice theories also share an emphasis on the material embedding of human activity, both as embodied in the human form of life, and as interconnected with natural and man-made material forms. Thus, to summarize again with Schatzki, "[a] central core … of practice theorists conceives of practices as embodied, materially mediated arrays of human activity centrally organized around shared practical understanding", and they conceive of social domains as the "total nexus of interconnected human practices" (2000: 11).

The practice approach has emerged as a significant framework in science studies, where the achievements of science are viewed as emergent from the social field of scientific practices (see for example PratiScienS, 2007). The general motivation for the adoption of the practice framework is to counter the positivistic view that science consists of propositional representations, and the making of science involves finding logical relationships among concepts expressed in propositions and ensuring that they entertain a correspondence relationship with reality. Knorr-Cetina (1999) and Nersessian (2005) both rely on the practice framework to theorize about epistemic change in the sciences, showing how the production of scientific knowledge is produced by the materially

embedded activities of the laboratory. A similar motivation is underlying my own adoption of the practice approach, with the difference that I am looking at how cognitive-epistemic practices inform novel technologies.

Epistemic practice constitutes the area where the practice approach meets distributed cognition. DCog and the practice approach in science equally suggest that knowledge-related practices unfold in time, and they span participants and material artifacts in a social setting. DCog differs from the practice framework insofar as it attempts to anchor its findings in the cognitive science tradition. As Nersessian states:

> "the practices uncovered are examined through a cognitive 'lens', i.e. in light of cognitive science investigations of similar practices in both ordinary and in scientific circumstances" (2002a: 135).

One advantage of this approach is that the findings can be compared against an existing body of research, and they have to be proven realistic in light of prevailing understandings about the nature of cognition and the mind. Meanwhile, the approach also challenges traditional cognitive science, elaborating its conceptual tools in new directions in light of empirical data from situated socio-cultural settings. In my own research, I will pursue the conversation with cognitive science, and specifically cognitively oriented studies of design, to further insight for building emergent concepts in the analysis.

The concept of 'space' or 'problem space' provides a good example of the anchoring I am talking about. Problem space has become an established metaphor for theorizing problem-solving within cognitive science as the set of logically acceptable solutions to a problem. Simon (1956) has for example described bounded rationality and satisficing in design as a process of limited search within the problem space, which is

"satisficed" by the first solution that is good enough, instead of spelling out all solutions and comparing them systematically against each other. Simon's account of problem space was formulated within the traditional framework of cognitive science, which limits cognition to the mind of an individual. Nersessian and her colleagues (Nersessian, Kurz-Milcke, Newstetter, Davies, 2003) have expanded the concept to comprise the cognitive, material and social configuration of the laboratory as an evolving problem space, which continually reconfigures itself as the research moves along. In my own research, the expansion of "problem space" has emerged as an early insight for capturing the generative nature of design thinking. Expansive spaces were found to be projected by new technologies as a space of opportunities, and then gradually charted through the formulation of tentative possibilities of novel user experiences.

I have argued that DCog and the epistemic practices approach are essentially congruent. I will now qualify this statement by adding that nevertheless, they represent distinct traditions of inquiry, which involve a difference in focus. Thus, while Hutchins' formulation of DCog is entirely compatible with large-scale versions of distribution, like a research laboratory, DCog has been typically applied on smaller scales, such as for the design of cognitive artifacts (Hutchins, 1995b; Hollan et al., 2000). The epistemic practices approach has on the other hand tended to define itself on a larger scale, emphasizing social and temporal distribution over the cognitive details of the process (see for example Knorr-Cetina's work). Thus, to mark my connection to both of these traditions, I prefer to use the term 'cognitive-epistemic practices'. When I want to make a specific reference to either one or the other tradition, I will talk about cognitive practices

and epistemic practices. I will use the term 'practice framework' to refer to the combined framework of cognitive and epistemic practices.

2.6. The research questions addressed by the research

The goal of the dissertation research has been to contribute to a socio-cognitive theoretical account of the processes of innovation in software that have emerged at the historical convergence of the social configuration of the project and the socio-technical context of information infrastructures. Open-source software development exhibits the features of this innovative configuration in a prototypical way, while embracing radical forms of open access and transparency, which make open source projects ideal sites of research. After an exploratory study of infrastructural software projects, the large-scale educational project of the Sakai Open Academic Environment (abbreviated from now on as S/OAE) was chosen as the site of the research. S/OAE has been created through online collaboration within an institutional open source community, and the research has relied on the archives of the collaboration, openly available to the public. The research has been seeking a theoretical understanding of the cognitive-epistemic foundations of social participation in the development of software systems, pursuing the following research questions:

1. What are the cognitive-epistemic practices that mediate the technical and social context in the evolving software system?

2. How are cognitive-epistemic practices correlated with social constituencies of design?

2.1. What is the distribution of the cognitive-epistemic skills that underlie these different cognitive-epistemic practices across social constituencies?

2.2. What is the role of tentative artifacts in the cognitive-epistemic practices of the different social constituencies?

3. What is the distribution of the cognitive-epistemic practices in the temporal process of shaping new software, how do practices associated with different social constituents supplement, follow, interface and facilitate each other as they inform the evolving artifact?

CHAPTER 3. METHODOLOGY

3.1. General considerations behind research design

The purpose of this chapter is to provide an account of the activities that have been undertaken to pursue the research questions, and to describe how credibility has been approached. The research has followed a grounded theory methodology guided by the framework of distributed cognition. The question of the scientific credibility or the soundness of research in the social domain has been a controversial area, especially with regards to the status of naturalistic inquiry and interpretive approach. I share the view of those who argue (Silverman, 2006; Morse, Barrett, Mayan, Olson, Spiers, 2008; Corbin & Strauss, 1990; Kirk & Miller, 1988) that the criteria of validity and reliability, generally used to appraise the robustness of scientific procedure, can also guide research in the social domain, and I will discuss my own efforts within this framework.

3.1.1. Validity

Validity talks to the way a relationship is established between data and theory, and prompts us to look at whether the conclusions obtained are responsive to the data in a way that is truthful and adequate to our concerns in the research. Two threads of ideas appear in discussions about the validity of social research (Cresswell & Miller, 2000; Donmoyer, 2001; Cho & Trent, 2006; Morse et al., 2008). The first aspect concerns the congruence between the research problem and the method, and the second describes the strategy for obtaining theory from the data. Both of them are equally important, but specific texts often restrict themselves to one or the other of the two.

Congruence between the research problem and method centers around questions of ontology and epistemology: what is the texture of the phenomena that we need to engage with in order to address the research problems we are studying, and how they are knowable for our purposes. I have been addressing these questions previously in discussing the research framework, where I was describing what I meant by the concepts of cognitive practices, skills and context, and how the research will approach them. The online setting also requires a careful rethinking of the methods used in investigating distributed and cognitive practices. The corresponding section will show how the framework informs research questions, and describes the methods for addressing the conceptual framework in the online context.

The second thread in validity is related to the strategy of obtaining theory from data. My research follows grounded theory, which is a qualitative methodology for the social sciences. While grounded theory has been established as a valid abductive strategy for building theory in the social domain, its outline is similar to qualitative research methods, which for this reason constitute a valuable additional resource for a grounded theory approach. The central ideas that inform grounded theory, notably the interpretative stance and the iterative nature of data collection and theorizing have also described as characteristics of qualitative social research in general (Miles & Huberman, 1984; Morse et al., 2008).

My take on grounded theory follows the Corbin and Strauss approach (2007), and looks at the methodology as a flexible procedure for building theory grounded in empirical data, which should be applied in conversation with the research terrain and the evolving theory. As Charmaz (2006) has put it, not only theory, but methodology should

be understood as emergent in a grounded theory approach. While Corbin and Strauss and Charmaz have been vocal about an understanding of grounded theory as a general methodological procedure, their illustrations have been often informed by symbolic interactionism, their conceptual framework of choice. In particular, two strategies central to achieving validity in interpretive theory building, the comparative method and triangulation, have been described with reference to the framework of symbolic interactionism. Related to this I will also show below how grounded theory can be applied in combination with the conceptual framework of distributed cognition and practices.

3.1.2. Reliability

The second notion of scientific credibility, reliability, has been described as the consistency of the methods and procedures used in research (Miles & Huberman, 1984; Corbin & Strauss, 1990; 2007). In connection with qualitative social research, Kirk and Miller have defined reliability as

> "the degree to which the finding is independent of accidental circumstances of the research" (1986: 20).

Grounded theory provides a rigorous outline for pursuing theory building, while also allowing for flexibility in the execution. In this context, reliability should be seen as clarity and transparency of method, which may be achieved by careful documentation of process and the rationales behind it, including an account of changes in plans. Related to my discussion of the strategies used to obtain validity, I will provide a practical account of the details of the research.

3.2. Outline of the ethnographically-informed cognitive-historical method

3.2.1. Characteristics of practice studies

How do we study practices, and how do we know what cognitive-epistemic practices are for the purpose of identifying them? First of all it has to be said that practices are not some kind of entities for the existence of which we want to claim proof. The 'practice lens' (Orlikowski, 2000) is a way of looking at phenomena, which is itself rooted in a phenomenological understanding of existence as process that unfolds in space and time, and a human knower whose experience of existence in space and time is mediated by the body. The empirical approach is correlated with this framework. Researchers pursuing practice studies (Hutchins, 1995a; Goodwin, 1995; Knorr-Cetina, 1999; Nersessian et al., 2003; Orlikowski, 2000; Lave, 1988; Vertesi, 2012) designed their methodology to allow for the observation of situated human activity unfolding over time, and to specifically support an engagement with its material as well as symbolic facets.

The specificities of ethnographically informed practice studies may be described by the following:

1. Most importantly, practice studies consider *activities as procedural sequences* unfolding over time, which can be captured by narratives.
2. Secondly, practices are viewed as *materially grounded activities*, and the approach is attentive to tools, documents, drawings, diagrams or other visual representations, as well as the broader environment as they are implicated in human activities.
3. Thirdly, materially grounded activities are viewed as *meaningful activities*, but unlike in other interpretive approaches inspired by symbolic interactionism or phenomenology, interpretations are not necessarily sought from a vantage point rooted in the immediate

concerns of those providing the meanings. Research is instead seen as a sense-making process which seeks to synthesize across polyphonic threads of meanings, and besides people's talk, observers continually make sense of the tools and context of the activities. Besides creating literal transcriptions from audio and video recordings, researchers also take detailed observation notes, which can be visited as data in the analysis. We may also seek access to relevant meanings from descriptions of activities, for example from interviews, or from locally produced archives of similar descriptions.

4. Finally, practice studies consider human practices as inseparable from *knowledgeable actors*: if persons are able to engage in practices, it is because they have acquired the embodied support of skills to do so. Practices are viewed as observable pointers to embodied capabilities for cognitive-epistemic functioning on the one hand, and to a history of their "acquisition" on the other (Hutchins, 1996; Bourdieu, 1977; Giddens, 1984). While neither the actual embodied skills, nor the historical process can be directly observed, systematic differences in the performance of practices support inferences about underlying skills and processes of acquisition. These inferences may be reinforced through secondary sources of data, such as career histories appearing in personal interviews or in public documents.

In sum, practice studies rely on fragmentary records from both direct observations and second-hand accounts of human activities, which are treated as meaningful data that converge on a narrative. Knorr-Cetina, for example, describes the materials she obtained from her ethnographic work and the work of narrative synthesis as follows:

> "We also scored by relying on *tape recordings (of meetings, shop talk*, etc.). […] Besides installing a machine-produced memory one can, of course, install

> informants as helpers [...] For example, physicists maintain a life record of the development of an experiment by sending to all institutes brief *e-mail summaries of meeting discussions and results* – a record that is fragmentary […] but nonetheless, on a technical level, better than anything even a group of observers could maintain. Here too we were lucky – all records of this sort plus all *meeting transparencies, internal notes, versions of talks and papers*, etc., were made available to us from the start, including […] materials not usually available in the public record. In addition, a great many physicists made themselves available over the years for lengthy *interviews* […]." (1999: 21-22, my emphases)

The output is characterized as a multi-threaded story with a serendipitous plot-line, which is then told from different thematic vantage points, in a fragmentary, kaleidoscopic approach:

> "The task of seeing through the thick growth of experimental manipulations in the hope of finding the *cultural switchboard that sets the direction of the project* is overwhelming. […] The first limitation concerns the *historical structure* of the work observed – characterized by a set of *ongoing stories*, without a clear *beginning* and an *ending*, perhaps, but with something like a *plot development*. Collaborations go through several *stages* whose full story add to the interest. Detectors, until they are built, involve many technical options, and a 'sifting out' *process that should be narrated*. The *story* of the search for a particular particle runs through an experiment from its earliest beginnings, vexing it to the very end. […] the *story lines* in the field were also frequently broken; many times, events did not march in step, sometimes last things came first, and the simultaneity of

events was overwhelming. [The present analysis] rather is kaleidoscopic. I look at conjunctions of activities by means of a succession of shifts in focus, as someone might turn a kaleidoscope to view various aspects of the empirical machineries, the technological machineries, and the social machineries of two epistemic cultures." (1999: 24, my emphases)

While practice-based methods can be described as ethnographic , they do not espouse the theoretical baggage of traditional ethnography. For this reason, I prefer to follow Nersessian (2005) in describing my approach as ethnographically informed cognitive-historical method. The cognitive-historical method was originally used to study how in the past individual scientists created and communicated novel theories, on the basis of cognitive resources available to them in particular scholarly cultures of thinking. This approach was relying on archival materials left behind by the scientists, historical accounts of the scholarly context, and present-day reconstruction of thinking processes grounded in the findings of cognitive research. It was subsequently adopted for the study of ongoing collaborative scientific activities, which have invited the use of ethnographic methods. Nersessian's characterization highlights the temporal, social and cognitive orientation of the study, and allows me to avoid a confusion with online ethnography (Garcia, Standlee, Bechkoff, Cui, 2009; Hine 2000), which has grown to be a method on its own right with a focus on researcher participation.

3.2.2. Applying practice-based methods to online practices

The development process I have chosen for my own study predominantly takes place in a non-collocated manner, through the mediation of online communication technologies. Because of the centrality of situatedness and physical presence in space in the practice

framework, the study of online practices requires a careful rethinking of method.[b] To illustrate the nature of the challenge, I will now turn to the activities in the development process, and assess them from the perspective of the practice framework.

S/OAE has been created by geographically dispersed contributors, whose primary mode of collaboration took place on various online communication platforms. The bulk of the development activity has been carried out on two platforms: the Sakai Confluence, which is a wiki, and the Jira, which is an activity tracker. Participants also frequently used e-mail to share and discuss ideas. Coders used code-repositories, such as github to share code, and functioning user interface is made available from local servers that coders have access to. IRC, a text-based chat interface was used by a small number of developers to keep in touch throughout day-to-day development, and to exchange quick questions and answers related to practical problems. Some appointed or informal leaders in the community kept blogs, which were generally used as outlets for personal opinions. The different platforms were interconnected through links, their functional demarcation was laissez-faire and loose. I have observed heated discussions emerge on the Confluence, Jira or e-mail, and even spread from one platform to the other. Everything shared on these platforms was publicly available, recorded, archived and searchable. Indeed, transparency was a central value of the community, and there was regular documentation of the activities that took place outside of this public realm, which included conference calls, workshops, and a yearly Sakai conference. Taken together, the

[b] While the method called online ethnography has received significant scholarly attention, it appears in many ways as a continuation of the theoretical heir of traditional ethnography, with an emphasis on researcher participation and understanding through active involvement. Therefore, it is not applicable to a practice-based approach.

platforms constitute a *site*, where the locus of activity may shift and spread between different local contexts.

The bulk of collaboration around Sakai OAE has been mediated by contents within online platforms. One way of conceptualizing the flow of activities as participants took turns in editing the online content is to think of so many lonely interactions with the computer in so many real world physical contexts: a person in an office, a chair, a table, a keyboard, a display with multimedia content. In this view, content is referencing practices in the offline world. Alternatively, we may also understand the situation as enframed by the online platforms of content, and bracket the real world for the purpose of research. This view suggests that shared content may be understood as identical with the practices. The latter approach appears preferable in a situation where participants typically appear to each other mediated by content, and the theorizing does not pertain to the embodied nature of the activities.

Knorr-Cetina has suggested a similar interpretation of online practices in her analysis of electronic financial markets:

> "Most of our world notions imply that the world is a place (however extended) or perhaps a totality of objects (e.g. the physical universe) 'wherein' we live, and 'in' which factual (e.g. globalization) and symbolic processes can be said to take place. The defining characteristic of this sort of world is that it is given or presupposed. In a timeworld or flowworld of the sort I will specify the content itself is processual—a 'melt' of material that is continually in flux, and that exists only as it is being projected forward and calls forth participants' reactions and contribution to the flux. Only 'frames', it would seem, for example, the frames

> that computer screens represent in a global financial market, are presupposed in this flowworld. The content, the entire constellation of things that pass as the referential context wherein some action takes place, is not separate from the totality of ongoing activities." (1994: 40)

Here is how a Sakai contributor described this experience:

> "I used to hold a respectable post at a respectable research institution, and would often be found next to actual people at a place where you could probably find me on any given day. I still hold that respectable post, technically, but the rest of it is out. The physical world is just for making sure I'm getting enough sustenance and charging my laptop for same, but the real world is the place where I am my avatar. Or at least it feels that way until I find my new rhythm."[1]

My online study has been looking at online content as the domain of practices. The available materials also contained ample descriptive information about face-to-face events and conference calls. Groups pursuing weekly or regular conference call discussions have been found to be using platforms like Etherpad and Titanpad, which provide immediate public screen sharing of text-based documents and make available a public archive of their full creation history. These services have been purposefully used by community members to self-document and share their work, in the name of transparency. The self-documenting archives belong to both the physical world and what Knorr-Cetina has called the online "flowworld". Insofar as they referenced events in the physical world, I have treated them as accounts rather than as actual practices.

I have also consulted a set of publicly available materials about the formal background and history of the Sakai Foundation, including the official public portrayal of

the Sakai community on its site, and a book-length history of its creation and early approach to development written by one of the founders and first executive director of the Sakai Foundation (Severance, 2011).

Finally, the research has focused on uncovering patterns of activities that pointed toward differences in underlying skills, and it has attempted to relate sets of related skills to wider communities of practice, such as professions. For this purpose, I have used the self-identification of participants and the labels they applied to their work, and I have compared the observed practices to educational and scholarly accounts in the professions.

3.3. Data collection

In the above I have shown how the framework of cognitive-epistemic practices may be understood as a conceptual lens for dealing with human activity as situated process. Unlike other types of social research, this approach does not rely on identifying categories of entities to systematically filter its sources of information, but appeals to the spatio-temporal logic of sites for scoping the research. Thus, Hutchins (1995a; 1996) identified the navy ship nearing port as the site wherein the distributed cognitive processes of navigation could be observed, and focused on the quartermaster's cabin and the bearing-takers' stations within this setting. Nersessian's research on distributed scientific practices in biomedical engineering considered specific university-based research laboratories over time (Nersessian et al., 2003), and Knorr-Cetina studied several large-scale projects within labs outside of universities (1999).

My research has followed the software system of S/OAE as it evolves through a diversity of forms. This evolution was crisscrossing between different online communication and collaboration platforms customarily used within the Sakai

community, which together can be described as the *site* of Sakai activities. Local spurts of activity adopted their own style of communicative work practices, creating a more circumscribed *locale* of activities. My research has focused on a set of such *locales*. The Confluence spaces were the primary sites of collaboration. In particular, Confluence spaces appeared as the sites where the coherence of content-mediated practices is routinely created. Thus, for the sake of simplicity, I have described these locales below in Table 3.1. by the URL of their Confluence space, acting as a directory or an orientation page.

The selection of the online locales included in the study corresponded to the research goal of following the evolving artifact. The selection process was relying on an initial effort of tracing the history of the S/OAE system: the most current format of the system was highlighted by the site of the Foundation, and I was working backwards from here. This effort of historical reconstruction was relying on directed searches, tracing hyperlinks and a systematic parsing of the list of pointers created by participants to showcase their work toward the community.

A major related challenge has been to establish the beginning and the end of the evolutionary process, since the research is centrally concerned with describing the tentative forms of the artifact and the nature of the processes that inform it. The temporal scope of the study has been established through a process described as theoretical sampling (Corbin & Strauss, 2007), where materials are sequentially drawn into the study to address research questions, until loose ends are tied up. I will provide a principled account of theoretical sampling in more detail related to data analysis, and here I will focus on the actual process. Initially, the temporal scope of the new Sakai was established

through the explicit labels for it: Sakai 3, 3AKAI, Nakamura (previously also called K2, the kernel of the system), and the Sakai Open Academic Environment, or Sakai OAE. Meanwhile, it became clear that the first descriptions tagged as Sakai 3 involved a polished idea of what the system should be like, including references to mockups, kernel technologies, as well as elaborate rationales in support of them. It became clear that the Sakai 3 project had a pre-history of early formation, during which a sense of the necessity for reinventing content management emerged among participants. The realization led me to embark on a targeted search for possible inspirations for the new Sakai. I was looking for sites where collaboration related to content may have occurred in the period preceding the official announcement of the development. The process resembled the detective work of seeking out hunches and following traces, and it was greatly helped by the discovery of an insider account which explicitly traced Sakai 3 back to a site of collaboration for improving the content experience within Sakai 2. Eventually, the creation of the related Confluence space was chosen as the start date of the observations, and the first official release as the end date. With this, the observations will span the period between January 2006 and October 2011.

Confluence spaces

A set of Confluence spaces were identified as the primary locales of the online study, and the research looked at the full range of wiki content within these spaces. These Confluence spaces constituted the core corpus of content that would be studied. I created java code to parse them into a database that covered all data available related to the authoring of the specific pages. Parsing of the core corpus identified the following dimensions of Confluence spaces within the core corpus:

Table 3.1: Overview of content in the core corpus

Identifier *	Wiki sessions **	Pages	Attach-ments	Project	Description
RES	545	266	186	Resources	Deals with user experience and tools focused on content management in Sakai 2.
SAKDEV + tags	255	150	1	Content	Deals with problems of content management in Sakai 2.
GROUPS	328	17	305	Groups	Exploration of groups
UX	407	29	336	OAE-UX	Exploration of groups
3AK	2004	1648	673	OAE	Deals with the development of the new Sakai.
TOTAL	3539	2110	1501		

*Each of these Confluence spaces can be accessed online by appending the identifier to the end of the URL of the Sakai confluence, described by the syntax: https://confluence.sakaiproject.org/display/[identifier] An exception is the SAKDEV space, where the locale of interest emerged as a subspace within the larger SAKDEV space, and it was delineated by using tags (contauthreq, referring to Requirements on content authoring and sakaiauthoring08, referring to a Workshop around authoring in September 2008).

**Wiki session refers to text-based contributions to specific wiki pages from different individuals, and may range from a few words to several thousand characters of texts, while attachments are typically pictures or document files.

***The character count is a rough approximation, and may be slightly inflated because of the presence of wiki formatting characters. The page count has been obtained by dividing the character count by 3000, a customarily used character count for single pages.

The SAKDEV space was not automatically parsed, and thus no character count is available. The page number given is an estimate.

Content in other platforms will be pulled in as deemed relevant for establishing the evolution of the artifact, following the logic of theoretical sampling (Corbin & Strauss, 2007).

E-mail lists

Google lists, available at http://groups.google.com/group/...

S/OAE-related: sakai-nakamura-tracker, 3akai, oae-dev; ***other***: sakai-kernel, sakai-dev, sakai-user, sakai-announcements

Collab e-mail lists, archived at http://collab.sakaiproject.org/pipermail/... and at nabble.com

S/OAE-related: oae-dev, oae-production, oae-urg, minispec-review, tl-lenses; ***other:*** sakai-ui-dev, announcements, production, sakai-ux, management

Jira spaces

Available at https://jira.sakaiproject.org/browse/

SAKDEV/Sakai+3+Roadmap, SAKIIIDESIGN, SAKIIIQA, REQ, OAEBLDR, KERNDOC, KERN, SAKIII

Code repository (Github)

Available at http://github.com/sakaiproject/3akai-ux

Blogs

Available at http://planetsakai.org/

Google documents

Referenced by individual links, and publicly accessible from GoogleDoc.

Conference call memos

http://etherpad.osuosl.org/oae-trg

http://etherpad.osuosl.org/s3

http://titanpad.com/oae-notes

http://titanpad.com/tl-lenses

http://tinyurl.com/tl-oae

3.4. Data analysis

In the analysis, I have followed a grounded theory approach. Grounded theory has been formulated to guide qualitative research in the social domain toward the formulation of theoretical abstractions. It has been described as an inductive/abductive strategy, with original formulations emphasizing the inductive process of reaching interpretive abstractions from empirical data (Glaser & Strauss, 1967), and later versions making space for formative theoretical inputs at all stages of the research. The abductive approach to grounded theory acknowledges the importance of an initial theoretical framework, and encourages cross-checks with existing theories as the inductive process of grounded theory making proceeds (Strauss & Corbin, 2007; Charmaz, 2006). Throughout the various applications of the methodology it has also become clear that grounding the theorizing process in empirical data is not a uniform process, and grounded theory should be approached as an emergent method, where the specific design of the research unfolds in conversation with the data and the research problems (Charmaz, 2006; 2010). Thus, rather than describing a specific procedure, grounded theory provides a set of strategic principles that promote validity in the interpretive formulation of theory from data. In my discussion of how I have applied the strategy of grounded theory in my

own research, I will focus on three of these, the iterative nature of data collection and theorizing, the importance of comparisons and theoretical sampling.

3.4.1. Iteration between data collection and theorizing

Grounded theory is centered on the understanding that theories in the social domain are the result of a meaning-making process, where we rely on our human cognitive apparatus to make sense of available data in terms of more abstract concepts that are responsive to the research questions. Meaning-making itself has been described as central to everyday human cognition. Shore (1996) illustrates this process with a compelling story, describing how the unexpected murder of a village chief disrupts the taken-for-granted flow of everyday life in the village, and how the unease of senselessness sets off a torrent of speculations and rumors about the identity and motives of the murderer, bringing into motion distant areas of cultural knowledge. Grounded theory is geared toward making sure that the sense-making process remains rooted in the studied phenomena: while researchers need to follow their hunches and formulate tentative interpretations, they also need to make sure that they are not jumping to early conclusions, they are for example ready to systematically examine their emerging interpretations, and they remain sensitive to aspects of the data that are not accounted for. Research accordingly iterates between formulating tentative interpretations from available data (or hypotheses, as Charmaz (2006) suggests) and seeking out further encounters with the phenomena to weed out and refine the interpretations, and discover their connections. Beyond the general strategy of iteration, my research has relied on a combination of memo writing and categorization as the strategy for producing tentative interpretations, and on systematic comparisons and theoretical sampling in the refinement of interpretations. In the following I will show how

these strategies were used from the early stages of research. I will also describe how the outlook emerging from this early work shaped the subsequent part of the research process.

My first encounter with the Sakai community was related to a case study of participative structures in open source development. The study took place at the end of 2009, at the time when public presentations about starting the development of a new Sakai system were gearing up, and it was focusing on the formal background of institutional representation and community source governance in the Sakai Foundation. Meanwhile, for my dissertation I was interested in the contribution of non-technical participants to software development, and the intense collaboration around the new system caught my attention. My early sense-making was focused on how participants were attempting to shape the new system through diverse inputs, and I discovered two artifacts that attracted collaborative efforts: the kernel and the diagram called *Design lenses* (see Figure 6.7). As I was trying to understand how these artifacts had come about, I realized that the work on the technical kernel appeared to have considerably longer history than the functionally oriented work on the *Design lenses* document. I found this strange, since the expectation would be to start with what to build and why. The case of the kernel started me on a journey of sense-making, seeking to understand the shape of a process where software technology was deployed before the roadmap for functionality was discussed and decided. This early encounter shaped my initial questions that I brought to the study of S/OAE development. What is the kernel? Is it really technical? What are the forms that developers engage with when they work on the kernel? Is it really so separate and disconnected from the *Design lenses*, and if so, why? And what is

the nature of the *Design lenses* work then? Was the kernel really preceding the *Design lenses*? When was work on each of those artifacts started? When eventually my decision was made to start a grounded theory study of S/OAE, these and similar questions were informing my initial interpretive efforts.

As I was making sense of the development process, it became increasingly clear that the labyrinthine world of the online Sakai community requires a systematic strategy for establishing the timeline of individual contributions. I designed two ways of dealing with this problem. The first was to display individual contributions to the wiki in a way that captures their original context. The second was the creation of a timeline of the evolution of the artifact.

Preparing the wiki pages for reading and analysis has followed a two-pronged display strategy. On the one hand, I created a database that aggregates saved wiki edits into individual sessions. For each session, the author and the time-stamp were recorded, and each wiki page could thus be summarized by a row of subsequent individual sessions (see Table 3.2). On the other hand, the actual contributions had to be displayed in the context. I opted to flatten out the criss-crossing layers of additions in a table, where contributions were listed in the order they occur in the document, alongside a number of the session they belong to (see Table 3.3). Reading the two tables together allows the reconstruction of the growth of the individual page through individual contributions. As a routine procedure, I prepare each wiki page by inserting the sessions database as a header of the document, and copy-pasting individual contributions paragraph by paragraph into a table. While the research is not relying on interviews, this preparation process may be seen as parallel to transcription. The process highlights the importance of creating

displays of data to facilitate the sense-making process, a step which has remained implicit in discussions of grounded theory.

Table 3.2: Example of data display created for a Confluence page[2]

Database id	Space	Confluence code	Session	Timestamp	Author
10508	SAKDEV	79857153	13	9/23/2008 14:02	Peter A. Knoop
10509	SAKDEV	16221583	12	9/23/2008 12:32	Mathieu Plourde
10510	SAKDEV	16221598	11	9/16/2008 10:39	Mathieu Plourde
10511	SAKDEV	16221658	10	9/15/2008 18:14	Clay Fenlason
10512	SAKDEV	16221631	9	9/12/2008 13:16	Noah Botimer
10513	SAKDEV	16221116	8	9/12/2008 13:09	Erica Ackerman
10514	SAKDEV	16221115	7	9/12/2008 12:27	Noah Botimer
10515	SAKDEV	16221114	6	9/12/2008 12:10	Mathieu Plourde
10516	SAKDEV	16221113	5	9/12/2008 2:57	John Norman
10517	SAKDEV	16221124	4	9/11/2008 13:51	Erica Ackerman
10518	SAKDEV	16221126	3	9/9/2008 13:31	Jon Dunn
10519	SAKDEV	16221766	2	9/5/2008 4:04	Michael Korcuska
10520	SAKDEV	16221851	1	9/2/2008 8:13	Mark J. Norton

Table 3.3: Example of data display created for the same Confluence page

Session	Paragraph number	Document	Codes	Artifacts, entities mentioned	Notes
1	1	The following Use Case domains are created for discussion at the First Authoring Summit. More specific use cases should be included as sub-sections within them.			
1	2	**Roles** (feel free to add more if needed)			
1	3	Linda Walsh, professor of chemistry James Smith, instructor, introductory computer languages			
4	4	Nick, professor of history			
6	5	Jonas Oldtimehr, retired, part-time online instructor			
1	6	Eizabeth Cantor, high school math teacher Ling Wu, graduate student Carlos Ramirez, undergraduate student John Hanson, IS administrator			
4	7	Celeste, subject librarian for history Lucy, government documents librarian			
8	8	Alex, grad student in Early Childhood Education			
1	9	**1. Site pages (CARET Portal, Anthony & Josh's tool)**			
1	10	Independent of how sites are organized (existing portal, GoogleTools, etc.),			
5	10	Independent of how sites are organized (existing portal, GoogleTools, etc.),			
1	11	there is a need to allow users to easily create pages that can be linked to other pages. These should be as flexible as possible in terms of layout and allow a rich set of content elements to be added to a page, including: marked up text, lists, tables, media objects (video, audio, flash, applets, etc.), and embeddable tools (GoogleTools, Widgets, synoptic Sakai tools, etc.).			
etc.					

The other display strategy was created to make the temporal context of contributions easily accessible. The first version of the table display contained the data, the authors and the platform of the contributions, but it quickly turned into a synoptic overview of the evolution of the artifact, with additional entries describing the tentative form and its place within a line of evolution (see Table 3.4). Labels were also added to indicate threads of the development process. The establishment of the temporal succession of the tentative forms of different parts of the S/OAE system has become a

central activity, with intense open coding taking place on the timeline itself. The approach has also provided partial answers to the initial questions about precedence. In particular, it was able to show a sudden onslaught of activity related to the two artifacts I was exploring, and made it clear that both the kernel and the *Design lenses* appeared in relatively finished forms, which pointed toward earlier phases of activity on different sites. The realization resulted in a search for precedents, and the temporal aligning of different forms has been an invaluable tool in this "detective work". The timeline is another example of the contribution of display to open coding. The comparative, synoptic arrangement was instrumental for coding the tentative forms of S/OAE and their relationships, and the summary approach has created a level of coding which has turned out to be more appropriate for the first phase of the study.

The temporal perspective has also facilitated the emergence of a narrative approach to the individual documents. While initially I focused on low-level open coding, the writing of memos has gradually become the primary means of data reduction. The temporal view contributed to an outlook on memos as complimentary to the synoptic display on the one hand, and the individual wiki pages on the other. This narrative framework became central in the sense-making process, helping to situate and process individual wiki pages within a wider process.

Table 3.4: Excerpt of the timeline used for keeping track of the evolution of S/OAE

Thread	Who	Date	Medium	Form of artifact	Source	Notes – contains, refers, resolved
Resources	Kristol Hancock, IUPUI	2005-10-03	Document	UI mockups, requirements document	2005-10-03 - 16613508 - Resources-5.pdf.docx	
Resources		2006-01-10	confluence space/home page	confluence home page	2006-01-10 - 16613508 - RES – Home.docx	… will be resolved in OAE
Jackrabbit 0.9 release	Apache Foundation	2006-02-14	content repository platform		http://jackrabbit.apache.org/jackrabbit-history.html	First official release of Jackrabbit
Timed release of resources	Salwa Khan (Texas State)	2006-02-15	JIRA issue	task (preceeded by feature requests)	2006-02-15 - _SAK-4493 - Timed Release of documents.docx	Resolved: 13-Sep-2006 16:40 In Sakai 2.3 (see Roadmap)
Order of resources	Salwa Khan (Texas State)	2006-02-16	JIRA issue	task (preceeded by feature requests)	2006-02-16 - _SAK-4507 - Ability to customize order of Resources.docx	Resolved: 19-Sep-2006 09:11 In0020Sakai 2.3 (see Roadmap)
Section (group) awareness	Gonzalo Silverio (gsilver), UMich	2006-03-03	document	UI mockups, requirements document	2006-03-03 - _SAK-4223 - Make resources group-aware+resource-sections	2006-03-26 - _SAK-4223 - Make resources group-aware.docx
Section (group) awareness	Jim Eng, UMich	2006-03-26	JIRA issue	task	2006-03-26 - _SAK-4223 - Make resources group-aware.docx	Resolved: 07-Apr-2006
Attachment helper	Jim Eng, UMich	2006-03-26	JIRA issue	task	2006-03-26 - _SAK-4224 - File picker helper should allow attachment by links in velocity tools.docx	Resolved: 06-Apr-2009 See also 2006-03-27 - 16613500 - Attachment helper.docx
Attachment helper	Jim Eng, etc.	2006-03-27	confluence page	requirements wiki	2006-03-27 - 16613500 - Attachment helper.docx	
Attachment helper	Michelle Bejian Lotia	2006-03-28	confluence page	link to mockup	2006-03-28 - 16613503 - Example Design.docx	2006-03-27 - 16613500 - Attachment helper.docx
Attachment helper	Michelle Bejian Lotia	2006-03-28	document	mockup	2006-03-28 - 16613503 - Example Design+AttachmentsUI.pdf	2006-03-27 - 16613500 - Attachment helper.docx
Section (group) awareness	Jim Eng, UMich	2006-03-28	confluence page	requirements wiki	2006-03-28 - 16613506 - Group awareness.docx	2006-03-26 - _SAK-4223 - Make resources group-aware.docx
Limiting display to public	Jim Eng, UMich	2006-03-28	JIRA issue + confluence page	task, requirements wiki	2006-03-28 - 16613505 - Limiting display to public.docx	Resolved 26-May-2006
Citation	Jim Eng, UMich	2006-03-28	JIRA issue + confluence page	task, requirements wiki	2006-03-28 - 16613501 - Citations and Citation Lists	Resolved 04-Apr-2007

3.4.2. The comparative approach

The second strategy I would like to discuss is comparison. The comparative method is an important analytic strategy, which serves interpretation as well as the goal of achieving the saturation of theoretical concepts by creating dense connections with both empirical data and with other concepts in the emerging theory. Comparison appears as grounded theory's version of the strategy of triangulation, as it is based on the use of various vantage points of the same phenomena.

It should be added that comparing data against other sources is sometimes misunderstood as a strategy of looking for identical instances to establish the existence of a pattern or typical behavior. This approach is itself linked to a particular framing of social phenomena as recurrent patterns of behavior, which is foreign to the DCog and practice frameworks, and more generally to grounded theory. Charmaz (2006) cites Glaser on this issue as follows:

> "Saturation [of interpretations] is not seeing the same pattern over and over again. It is the conceptualization of comparisons of these incidents which yield different properties of the pattern, until no new properties of the pattern emerge. This yields the conceptual density that when integrated into hypotheses make up the body of the generated grounded theory with theoretical completeness." (Glaser, 2001: 191)

Miles and Huberman relate this interpretive strategy to the *modus operandi* process used by detectives:

> "When the detective amasses fingerprints, hair samples, alibis, and eyewitness accounts, a case is being made that presumably fits one suspect far better than

> others; the strategy is pattern matching, using several data together. Diagnosing engine failure or chest pain follows a similar approach. The signs presumable point to the same conclusion and/or rule out other conclusions." (1994: 267)

I have already shown how I used synoptic display to facilitate comparison across different contributions to the S/OAE system. The comparative approach has for example contributed to discovering different types of collaboration underlying the confluence pages, such as the planning hub or the collection page.

A second line of comparisons has involved the existing literature. While early formulations of grounded theory focused on empirical data at the expense of the existing field of research (Glaser & Strauss, 1967), later versions have explicitly acknowledged and encouraged conversation with theories (Charmaz, 2006, Corbin & Strauss, 2007). The cognitive-historical method seeks to establish that findings are realistic and relevant in light of current understandings of cognition, which is achieved through constant comparison with theoretical literature. My own data have been accordingly anchored in cognitive studies of design, and the broader theoretical context of creative thinking.

I have described earlier how the notion of the expansion of the design space has emerged at the juncture of my empirical material and the cognitive theory of design. In the attempt to make sense of software technologies and technical approaches in the S/OAE development, I have also reached out to relevant literature in the software domain. Thus, for example, the specification of Apache's JSR-170 standard defining content repositories as a service within system architecture, and its implementation as the Jackrabbit platform appear as significant inputs in early Sakai discussions about content. Contextualizing this component has involved the study of technical and release

documentation attached to the original releases, and an exploration of the professional literature of service architectures, which will continue alongside the study of S/OAE materials. Early engagement with the materials has also shown that collaboration revolves around professional practices rooted in the field of user experience design. The practices pursued at the Sakai site have been compared to accounts of professional practices within the user experience (UX) community, in particular to understand how these fit in with the broader software development process and what contributors may be trying to achieve through them. For example, contributors often used the terms use case and scenario for short stories that described ways of engaging with technology in the educational context. Their stories tended to resemble each other, but appeared to be more concise and containing less detail than many of the scenarios from UX practice. Eventually, I analyzed these stories as liberal local takes on a professional tool, which were guided by the local needs of the design process.

Finally, categorization was pursued as a comparative procedure. Data behind related categories was constantly pulled together for conceptually enriching and refining the analysis. To cite the example of use cases and scenarios again, a systematic comparison of their use has allowed me to explore to what extent they were rooted in systematic investigations of the educational domain, chance personal observations and imaginative recombination of general knowledge of the field.

3.4.3. Theoretical sampling

Similar to comparisons, the sampling strategy of grounded theory also serves the interpretive saturation of the emerging theory. Theoretical sampling is described as an approach to choosing empirical sources to support the needs of theory-building, and

saturation sampling is a variety of theoretical sampling where new data are drawn into the analysis to support the meaning-making process, until the point when a dense and internally coherent conceptual architecture is reached. Seeking out new sources of data may be related to blurry points and gaps in the interpretation of practices, or the need to dig deeper. Thus, the discrepancy between the kernel and the user experience thread discovered during my initial encounter with the Sakai community guided the collection of materials at the very beginning. As it became clear that each of these threads started with advanced forms of software, the sampling of materials was extended to earlier phases of the process, and specifically to formative phases happening before the new system was named. Engagement with the materials has also resulted in the abandonment of some of the original problems related to the notion of representation behind the organizational structure of the Sakai Foundation. Specifically, it was assumed that contributions from representatives of the different universities would be rooted in the aspirations of a wider body of constituents at the educational institutions. Understanding how the bridge was created between local collaboration and remote sites of motivation would have required a study of covert strategies, and the conducting of personal interviews. Familiarization with development settings has however shown that the links with the sending institutions were generally tenuous, and the shape of the development process could be understood without accessing the private perspective of contributors.

Earlier I have identified a core set of wiki spaces, which constituted the backbone of the study. These have been chosen to be included in the study because they exhibited the temporal evolution of S/OAE. Other sites of collaboration, such as e-mails, blogs or repositories of memos became included as needed to clarify aspects of the process. Most

important among these was a heavy reliance on the sakai-kernel and sakai-ux email lists, which proved to be more informative for the threads of work they covered than the related Confluence pages. The design of S/OAE took place in two relatively distinct phases, an early formative phase, when collaboration took place around emergent goals in emergent groups, and the subsequent phase of the managed project, which was characterized by process planning and established structures of decision-making. The research has found that formative inputs came during the first phase, and the second phase consisted of routine execution of those early insights. I was able to reach the saturation of interpretation in the analysis without a detailed study of these contributions.

CHAPTER 4. CREATING THE SPACE FOR INNOVATION IN DOMAIN-DRIVEN OPEN SOURCE

4.1. Introduction

Sakai defines itself as a global open source community created by institutions of higher education that develops open software for teaching, learning and research with the facilitation of the Sakai Foundation[c]. It represents a special case within the recent trend of organizational involvement in open source software (OSS) that I suggest to call domain-driven open source, because it creates a space for participants of an application domain to design, create and maintain their own software. The notion of domain-driven open source complements the traditional focus of open source on implementation with increased attention to the design of functionality.

Open source and its approach to software innovation has seen considerable research interest. In particular, open source development has been described as a novel approach to coordination by means of web-based collaboration platforms. Taking this account further I will suggest that the initiation of the new software platform under the code name of Sakai 3 may be accounted for as a distributed cognitive-epistemic process, in which participants were actively engaged in making space for collaborative innovation for and by the application domain. In the first part of the chapter, I will introduce the Sakai community and situate it within the open source landscape. In the second part, I will show how social governance, modular architecture and web collaboration platforms were used by Sakai participants to create the conditions of their own distributed epistemic

[c] In line with what's customary in the open source world, I will use the term 'Sakai' to refer to the open source community as well as the software project that brings them together. I will use 'Sakai Foundation' to refer to the organization behind Sakai.

work. I will argue that an account of innovation in the case of Sakai 3 needs to include a fourth element, of a cognitive-epistemic nature, and I will show how Sakai 3 came about through the creation of a design space, which served as a cognitive-epistemic framing for subsequent work on a new system.

4.2. Domain-driven open source and innovation

4.2.1. Building software for higher education from within

Open source software development became popularized by such widely-read works as Raymond's *The Cathedral and the Bazaar* (1999), Moody's account of Linux in *Rebel Code* (2001) and the collection *Open Sources:Voices from the Open Source Revolution* (Dibona, Stone, Ockman, 1999). These early accounts contributed to a popular image of OSS as the spontaneous collaboration of self-organizing individuals. Many accounts were based on the iconic cases of the Linux operating system and the Apache web server. Linux was started by Linus Torvalds as a hobby project based on his interest in creating a Unix-like operating system. An early version of the software was shared with the programmer community alongside the invitation to contribute. Torvalds remained responsible for integrating both new features and bug patches into subsequent releases of Linux. The Apache project was created after the developer of an openly distributed and popular web server had to abandon his work on the software, and a group of developers from around the world with a strong interest in using the web server decided to pick up the project, and organized their work processes around their distributed situation. In the Apache story, contributors were also employees, but their involvement has to this day remained at the level of individual volunteers, who do not act on behalf of their firms (Fielding, 1999).

Surveying a cross-section of diverse OSS projects, Gacek and Arief (2004) encountered a more nuanced picture, with many open-source projects involving only one person. They found that only two characteristics were shared across all projects that participants and others characterize as open source: they used an open source license and their contributors created software for their own use. These two features may thus be seen as constitutive of the open source phenomenon, while the term itself allows for considerable diversity in terms of approaches to collaboration.

Related to the latter point, studies have pointed out the growing involvement of firms in OSS. Fitzgerald (2002) coined the label Open Source 2.0 to highlight a new era around OSS dominated by firms rather than individual volunteers. A widely cited example is Mozilla, which started as proprietary software, and it was released as OSS under the Mozilla Foundation. The Mozilla Foundation today has final say in the direction the developments take, notably in the inclusion of new modules, and relies on full-time staff employed within the Foundation or at one of its umbrella organizations[d]. The employees organize and manage development efforts: they chart a roadmap of development and coordinate releases and maintenance (Mockus, Fielding, Herbsleb, 2002). Related to the involvement of firms Wasserman & Capra (2008) suggest to distinguish between community OSS and commercial OSS, characterizing the former as having no firm involvement.

In fact, organizations have also been involved in the world of community OSS. The fact that many developers work on OSS as employees has been widely cited. The Apache Group consisted of developers who were using the web server for their job. Also,

[d] Netscape was the founder of the Mozilla Foundation, and Sun is its more recent owner.

both the Apache and Linux projects have created organizations (the Apache Software Foundation and Linux Foundation) to provide continuity to the software beyond the specific developers.

In a survey of OSS projects Capra and his colleagues found that the involvement of firms takes a variety of forms (Capra, Francalanci, Merlo, Lamastra, 2009). In the *coding model*, firms contribute to the development of the software to gain specific business advantages: firms may lend their employees to work for the project during their working time, with the underlying motivation to gain an influential position and drive future developments (Dahlander & Wallin, 2006; Krishnamurty, 2003), they may hire and pay a developer already working on the project to develop specific functionalities[e], or they may donate proprietary code to the project, usually at the initiation. In the *support model*, firms provide resources to support a development project with a range of activities beyond coding, from marketing through testing to customer support: they may initiate the community and remain legally connected to it, as in the case of the Mozilla Foundation, or may step in as sponsors. Finally, in the *management model*, firms are only involved as project administrators and coordinators. This type of involvement may happen to initiate a community, bringing partners together and providing strategic direction to their work (Krishnamurty, 2003).

These various studies of the OSS 2.0 phenomenon have focused on the involvement of software firms. At the same time, organizations outside of software industry have also taken an interest in open source software, creating organizational

[e] Hars&Ou (2002) found 45% of contributors to be paid by firms, while Wasserman&Capra (2007) suggest that within the different projects 50% to 95% of code has been developed by paid contributors.

platforms for the collaboration of institutions to develop open source software from inside the application domain. This approach has been at times described as community source (Hanganu, 2008; Wheeler, 2010), but the term itself can be misleading, because it was originally coined to describe the practice of releasing code to a restricted community. Instead of community-source, I suggest to use the term domain-driven open source, to highlight the fact that these institutional collaborations develop software for their own use within their specific application domain.

The Sakai project started as the collaboration of academic institutions under a research grant from the Andrew W. Mellon Foundation to build an open source collaborative learning environment for higher education (the Sakai Collaborative Learning Environment, from now on referred to as Sakai 2) by combining software which had already existed at the institutions. As part of the grant activity, project members sought and found support from other institutions to create a Foundation for maintaining and improving the initial open source system. Other open source initiatives based on the collaboration of organizations include the Kuali Foundation (building administrative systems for education), the Open Source Portfolio Initiative (building a portfolio system for education), Jasig (building software support for systems in higher education, like uPortal for creating a portal architecture, and CAS, a single-sign on authentication solution), NIEM (building software in the areas of law enforcement and justice), and NHIN Connect (creating software for the sharing of medical records). The Apereo Foundation was created by Jasig and the Sakai Foundation in 2012 as an umbrella organization for open source initiatives in higher education. After the merger S/OAE was

overtaken by Apereo under the name Apereo Open Academic Environment (Apereo OAE), while Sakai Foundation kept the Sakai Collaborative Learning Environment.

As I have suggested, software development is not a primary source of revenue for the institutions in the Sakai Foundation, its value is perceived indirectly as the cost of support for their core activities. Many institutions behind Sakai admit that they have been influenced in their choice of open source by the state of the LMS market, and particularly the situation of lock-in with large vendors, such as Blackboard and WebCT (Severance, 2011). LMS providers have been known to make major technological decisions without consulting their clients, including the withdrawal of maintenance and support from older versions of the software. The remedy to this situation was the creation of a development community that remains guided by the strategic interests of the member institutions.

The fact that institutions develop software for their own use within the application domain does not imply that end users are directly involved in development. Sakai contributors are employees from instructional technology departments in higher education, delegated by the Sakai member institutions. In the following, I will use the terms contributor and participant as synonyms to refer to those people who took part in creating S/OAE.

The primary purpose of Sakai is to support instruction in the higher education context, and its primary user pool consists of students and instructors. More specifically, the community's stated goal is to support instructors. At the same time, Sakai 2 has been taken up for other uses, notably for academic and administrative collaboration at the institutions. In this quality, the user group has come to peripherally include administrative support personnel and people in research-related roles. The latter group shows significant

overlap with the primary user pool of instructors and students. Sakai does not explicitly rule out the possibility of end user involvement, and this possibility has become tangible in Apereo OAE in the form of support for user-developed widgets, but the Sakai Foundation itself has not provided specific arrangements which would involve a larger pool of end users in actual development. Table 4.1 provides an overview of the organizational ecosystem of the Sakai Foundation.

Table 4.1: Overview of the social-organizational ecosystem of the Sakai community

Sakai Foundation members (officially called Sakai Partners)	Higher education institutions Software companies
Sakai community contributors (will also be called participants)	Employees from instructional technology departments and software companies Consultants hired for a specific project
Directly supported end-user group	Instructors
Indirectly supported end-user group	Students
Other end user groups	Persons in researcher roles Administrative personnel

In sum, the Foundation's goal has been to bring together professionals from supporting roles, primarily from instructional technology departments within the schools, to build and maintain software that allows for instructors to teach and educate students. This goal was central in the perspective of S/OAE contributors.

4.2.2. Licensing

Open source projects revolve around the making and releasing of open source software. Open source licenses are legally binding documents that describe the conditions for the sharing, use and modification of the software. Sakai uses the Educational Community License, version 2.0 (ECL-2.0)[3], which is a derivative of the common Apache 2.0 license[4]. It allows others to freely download and use the software in any context, regardless of purpose. It has permissive stipulations for modification, allowing for both open source and proprietary reuse of parts or whole of the software. The ECL-2.0 license makes contributors' life worry-free in terms of their own contributions, while the issue of licensing tends to come up in connection with using source code from other sources, where the compatibility of the other license with ECL-2.0 has to be considered. Thus, when the option of including a rich text editor was considered, licenses had to be taken into consideration besides functionality. The back-end components chosen for the new kernel were also open source. In case of the adoption of open standards specifications, like JSR, the decision also considered the availability of open source implementations.

4.2.3. Innovation in open source development

Open source has also been characterized as a development process on its own right. Research has pointed out a common cycle of OSS development which consists of consecutive phases of code writing, pre-commit testing, beta release, debugging, production release, and maintenance. This development process has been accounted for as a characteristic approach to innovation, which favors incremental changes (Jørgensen, 2001). In this approach, new features arise slowly from bug reports and related user feedback, as described by the dictum '*Release early, release often*' (Raymond, 1999).

The initial phases of planning established in conventional software engineering (such as requirements specification, analysis and software design) (Sommerville, 2010), are collapsed into building the initial prototype – an activity not regulated by an established choreography, and usually performed by the single person or small group who follow their personal "itch" for creating the project.

The disappearance of an early phase of software design in open source appears problematic for domain-driven projects. Fitzgerald has argued that early phase activities become more significant for OSS 2.0 projects for a variety of reasons. While early community projects tended to work on network infrastructure software, such as the Apache web server or the Linux operating system, OSS 2.0 projects build more visible end-user applications, which are also based on complex and specialized domain knowledge external to software engineering. Domain-driven projects bring together stakeholders in the application domain, and if their goal is to build a new system, this implies a need for supporting a consensual approach to the initial conceptualizations of the system. As a result, these projects can be expected to spend more time with requirements, and install some related process. Noll (no date) surveys the processes that have been found to drive innovation in the evolution of OSS, and suggests that projects can systematically rely on the collection of user suggestions (Reis & de Mattos Fortes, 2002; Feller & Fitzgerald, 2002), follow commercial state of the art (Nichols & Twidale, 2003), and even do conventional requirements elicitation (Nichols & Twidale, 2003). At the same time, no approach has been specifically associated with the open source context, and understanding the difficulties and opportunities of defining novel software in the open source context appears as a central challenge that OSS 2.0 projects are facing.

Meanwhile, the landscape of open source innovation appears to be more varied than the canonical account of OSS development suggests. There are OSS initiatives that are focused on a single, evolving software, but even these may be organized in a modular, extensible way, which allows for the creation of new subprojects. Extensible platform-architectures outside of open source have been shown to act as drivers of innovation in software (Gawer & Cusumano, 2002; Evans, Hagiu, Schmalensee, 2006), and Fielding (2005) has suggested that Apache, Eclipse, Mozilla Firefox and the Linux kernel have been designed to invite novel contributions on the principle of extension. Some open source communities, like Apache, Moodle or Sakai, have fully embraced a multi-project approach, where the community serves as a hotbed for the incubation of novel open source projects.

The educational open source platform Moodle is based on a plugin architecture, where the central codebase is maintained in a more traditional development environment by the parent company, and besides providing bug patches, the open source community around Moodle can contribute plugins, which work as standalone open source projects. These "contrib" projects go through a validation process before being adopted into the Moodle plugin library.[5] The open source community around Moodle may contribute to the emergence of novel contrib ideas, but otherwise Moodle is like typical open source projects in that it has no guidance in place for the early phases of software development.

The Apache Foundation serves as an umbrella for a wide range of projects based on the Apache server, and it has rules of social governance in place for the creation of novel projects. These rules guide new projects (called podlings) through the early phase

of incubation, before they can prove their maturity to be formally accepted as an Apache project.[6]

As I will discuss in more detail below, Sakai gradually established a combination of the two approaches. The first Sakai system was created from software previously built at the participating universities, which resulted in a tool-based architecture. Each Sakai 2 release contains a set of core tools, which are part of the main release, and a series of contrib tools, which have been validated to work with the release. Thus, Sakai 2 accommodates new contributions according to the logic of modularity, like Moodle. Sakai has also formulated an Apache-like process with rules that guide projects through the early phase of innovation.

Besides endorsing a modular, multi-project approach, Sakai has been active in supporting a domain-driven innovation process by making space for discussions rooted in the application domain. To support processes of user-feedback central to the open source philosophy, Sakai has put in place extensive support for non-developer community discussion. These discussions contribute to domain-driven change in the system, and they have been specifically active in domain-driven innovation.

In light of the above, Sakai may be interpreted as an organizational design experiment in innovation which establishes a domain-driven, collaborative approach to the making of software. In the following I will show how Sakai was relying on the open source approach in the broad sense to create space for domain-driven innovation.

4.3. Making space for design

Beyond the specifics of a development process, open source has been described as an overarching approach to coordinating development on the web, which leaves considerable space for emergence of process. Raymond (1999) described this as the bazaar approach, contrasting open source with the top-down, managerial style of conventional software engineering, which he likened to cathedral building. In this view, open source does not look to top down management and strategic planning for coordinating the processes of work, it is instead characterized by the adoption of coordination tools that allow for local emergence of coordination.

Raymond's account downplays the possibility of accommodating traditional management approaches locally, as part of an overarching open source logic. Meanwhile, OSS 2.0-type projects have been documented to return to these approaches in one way or another. Fitzgerald (2006) suggests that the hallmark of OSS 2.0 is its reliance on strategic planning and more rigorous project management, as known from conventional software engineering. Mozilla's previously referenced development process, for example, involves strategic planning and considerable top-down involvement in managing releases and bug fixes (Mockus et al., 2002), but its overall approach is such that it invites voluntary contributions from software developers. Sakai also incorporated the managed project within its broader strategy for supporting innovation.

In my analysis of Sakai below I will argue that Raymond's bazaar account overlooks the active role of contributors in creating coordination locally, and I will show that coordination work can be interpreted as part of the overall cognitive-epistemic

process of design, wherein the participants work on reflexively creating the conditions of their own epistemic work.

Three aspects of open source development have been in particular discussed as contributing to the coordination of OSS work processes:

1. Open source approaches to governance
2. The modular architecture of open source software
3. The use of online collaboration platforms

In the analysis I will show that governance and architecture were used in a constitutive gesture, to create the overall context wherein the design of Sakai 3 could take place, whereas online collaboration platforms tended to be used to guide and facilitate design work locally.

I will also suggest that an account of innovation requires a fourth element of a distinctly cognitive-epistemic nature: the creation of a design space on the basis of a fragmentary and open-ended vision for a new Sakai, which invited cognitive-epistemic processes of meaning-making and conceptual construction.

4.3.1. Open source governance in Sakai: creating the conditions of epistemic collaboration

In terms of governance, Sakai went through three major phases of design efforts that were relevant for S/OAE. The first phase was related to the creation of the Foundation; it started at the time that the Foundation bylaws were drafted, and spread into most of its first year (end of 2004-end of 2005), under the leadership of Charles Severance, first as Chief Architect, and later as Executive Director. The second phase was connected to the era of Michael Korcuska as Executive Director (July 2007 to March 2010), who followed

community demand to spur actual contribution to code through a redesign of development practices. After redesigning routine processes of release management and maintenance (which I will not describe here), Sakai turned to reflecting on its innovative practices, and much of 2009 was spent with crafting, debating and establishing a framework for forward looking development, focusing as much on what to build (the traditional requirements phase) as how to build it. Critics at the time suggested that the effort was designing a process specifically for the slowly emerging vision of a new Sakai, which subsequently became S/OAE. The new framework defined a phase of project maturity, when projects were encouraged to rely on some form of management. Sakai 3 followed this path and entered a new phase of intensive project design in the second half of 2010. Besides setting up a managed project with an intricate structure of overseeing groups, the design involved the formulation of contracts for the partnering universities. This was a move which could be seen as repeating the initial creation of the Sakai Foundation, albeit on a smaller scale.

The first phase: Designing the Sakai Foundation

As the initial grant period was drawing to a close, participants in the grant project started working on the legal backgrounds of the new community (the bylaws and the license) as well as on ways of supporting community practices. For the latter purpose, active grant-time participants created the Sakai Community Practices (SCP) working group with the charter to document existing practices for the benefit of participants from post-grant period Sakai Partners. In this case, a core group was sharing its own practices with a wider community, which was in line with the Foundation's stated role to inform and guide contributors. Some tension was present in this work due to the fact that participants

wanted to avoid dictating new practices (hence the goal of documentation), but they did not want to perpetuate the work habits of the grant era, characterized by strategic planning, management and isolated groups. Their openly professed ideal was the Apache group, and they appeared to have found a middle ground in describing existing practices that were in line with this ideal, while acting slowly to change the others. Eventually, the documentation of some practices planned for was quietly dropped. Drafts of the software processes contained verbatim sections of Apache documents, referenced Apache on meritocracy, but also emphasized volunteering, the value of communication and transparency. More than any other guidelines, the latter three appear to have been central to the way work gets done in Sakai. I will note that these attitudes are equally valued in academia and OSS.

The SCP group started out with significant amount of formalism borrowed from Apache, talking about committees that design their charter, deliverables, milestones, decision rules and voting. They gradually moved toward a more informal approach to self-governance, which emphasizes the use of online infrastructures and documentation. The committee-related language was dropped, and project groups provided the scaffolding for work within Sakai. Projects were typically set up with a loose charter and a loose tracking of membership. No provisions were made for defining who can initiate projects and what their content may be. Following the legacy of the grant era, there were three type of project groups: discussion group, workgroup, and software project. Defining the role of project coordinator and subsequently hiring a core member to this Foundation position was instrumental for the project-based organizational approach. By setting up online workspaces, monitoring inactive sites of work, and keeping up constant

communication with individual projects, the project coordinator has been central to facilitating work in Sakai ever since.

Table 4.2: Overview of projects related to Sakai 3

	Focus	Start	End
Resources	user experience and tools of content management in Sakai 2	January 2006	August 2008
Kernel 1	separating a core set of back-end services within Sakai 2	June 2007	October 2008
MySakai	proof of concept implementation of dashboard UX for Sakai 2	fall 2007	February 2008
User Experience Improvement	improving the UX of Sakai 2	March 2008	December 2008
Content Authoring	exploring content structuring in Sakai 2	March 2008	December 2008
Kernel 2 (Nakamura)	back-end development of Sakai 3	fall 2008	past 1st release
3akai	R&D project for developing a UX prototype based on Kernel 1	January 2009	July 2009
Hybrid	Sakai 2 and 3 hybrid release	spring 2009	past 1st release
Simple Learning Environment	developing a UX prototype based on Kernel 2	August 2009	June 2010
Groups	exploration of what groups may be in the Sakai 3 user experience	June 2009	November 2009
Instructional visioning	collecting and organizing the teaching and learning capabilities	August 2009	past 1st release
Investigation	goal-directed research on assignments, with user interviews	September 2009	June 2010
Sakai OAE	S/OAE pre-release, S/OAE 1.0	July 2010	September 2012

Green: Projects leading up to Sakai 3

Blue: Central Sakai 3 projects

Orange: Sakai 3 side projects

The idea of a new Sakai gradually emerged out of a small number of such projects, most importantly Resources, which looked at ways of improving user experiences with content in Sakai 2, MySakai, a contrib-type development project, which created a proof of concept implementation of a widget-based, dashboard user interface for some of the existing Sakai 2 functionality, and the User Experience Improvement Initiative (UXI) for Sakai 2. Later, when Sakai 3 became established as a project on its own right, it was accompanied by a small number of satellite projects. Table 4.2 presents the projects that contributed to Sakai 3.

The second phase: Designing for growth

During its first year, Sakai was successful in terms of adoption, but there was a growing unease in the community related to the insufficient amount of coding contributions. Michael Korcuska, the new executive director of the Foundation, sought remedies through the design of new project structures. He initiated a process, wherein active participants of the community first designed the processes for routine development, and subsequently outlined a process for supporting novel contributions. Central to the latter was a detailed project lifecycle, which acknowledged the importance of unstructured and emergent "*R&D"* activity, and described three types of projects corresponding to three consecutive phases of development: an informal phase of "*incubation"*, with the goal of refining the project goals, the "*product development phase"*, which requires the acceptance of the Sakai Foundation Board on the basis of a document that highlights the relevance and feasibility of the proposed software, and the "*maintenance phase"* for the mature product. The new framework was originally designed by the Board, a core group of elected community leaders, and later debated and refined by the wider community.

Despite the intense community involvement and the tangible need within the community for better guidance from the Foundation, the top-down initiation of structures was unwelcome by some. It is not clear how much influence the framework had in general for new development across the community, but Sakai 3 was clearly following the path it described.

The first Sakai 3 project was formed under the name of 3akai parallel to the initiation of the framework at the beginning of 2009. 3akai was formally initiated in early January, and the Sakai Board released its recommendations for the new process in February. 3akai was recognized as an R&D "incubation project" once the process was formally accepted. It became an incubated project at the beginning of 2010, and entered the phase of product development with the creation of formal structures of development around a managed project in the fall.

The third phase: Setting up a managed project

Along the lines recommended in Korcuska's framework, Sakai 3 product development was designed as a managed project. Notions for structuring the development process under a project manager had been formulated as early as 2008, and after being discussed from time to time, a structured approach was adopted in 2010. This included a division of labor between UX design, client-side and server-side development teams, and a QA (testing) group. A contract was drawn up, in which participating universities agreed to fund the project. A Steering Group was also instituted for strategic decisions and oversight of the project, with one representative from each participating institution. In the early days of the managed project, the User Reference Group was initiated to strengthen

the requirements-related design of the new Sakai, and to oversee the implementation of planned features.

4.3.2. Reflecting on architecture for organizing epistemic work with software

Starting with early accounts of OSS (Torvalds, 1999; O'Reilly, 1999; Bollinger, Nelson, Turnbull, Self, 1999; Raymond 2001), it has often been stated that the distributed character of open-source development was made possible by a modular software architecture. Specifically, modular organization provides an integrative framework which allows for developers to work on different parts of the software in parallel, without the need to coordinate their changes through communication.

Modularity had emerged as a central concept in early software engineering theory. In 1972, Parnas formulated information hiding as a central criterion for achieving modularity. This principle, also known as encapsulation, states that a module only needs to share the nature of its external behavior defined in terms of an input-output interface, and the details of its internal operation, or how this behavior is achieved, should remain hidden. In a well-designed modular architecture, the internals of modules such as sub-routines and variables should not be accessed directly by other operations of the system, only indirectly, through an interface. This approach allows for modifying the way a module performs its operations without changing its connections to other modules – a phenomenon described as loose coupling.

While OSS has been leading the way in modular architectures, some have also suggested that the application of modularity was often lax. The proprietary system released as Mozilla was for example monolithic, and moved toward a more modular architecture in the first years of its open source life (MacCormack, Rusnak, Baldwin,

2006). Linux has been widely praised for its modular architecture, consisting of a kernel and various collections of modules and drivers, which made massively parallel development possible (Torvalds, 1999; Godfrey & Tu, 2000; Baldwin & Clark, 2006). Researchers have also pointed out that Linux's modularity has significant exceptions (MacCormack et al., 2006; Narduzzo & Rossi, 2004), such as an overreliance on kernel-related global variables (Schach et al., 2002; Yu, Chen, 2004). Looking at the division of labor around the Apache webserver, another highly modular system, Mockus, Fielding and Herbsleb (2002) found that there was no strict division of labor with single-module ownership and contribution. The above details suggest that the structure of code provides guidance for the general outlines of the division of labor in open source projects but contributors may find themselves trespassing in each other's areas at times. Modularity has also been perceived as a desirable outcome, (Milev, Muegge, Weiss, 2009; MacCormack et al., 2006), and OSS projects have spent considerable efforts in attempts to expand it. The latter suggests that modularity should be seen as a principle behind work in progress, which allows participants to reflect and act upon the conditions of their own development.

Some authors have suggested that it is extensibility rather than modularity which defines distributed collaboration in OSS. Scacchi (2005) has referred to software extension mechanisms, which invite novel contributions around a single platform, and Fielding (2005) has claimed that systems like Apache, Eclipse, Mozilla Firefox, the Linux kernel, or the World Wide Web establish a centrally controlled code base with interfaces designed to promote anarchic collaboration through extensions.

Modular and extensible architecture was important for the evolution of both Sakai systems. Sakai 2[7] was created from the integration of web-based software systems that had been developed at different universities within the United States[f]. The goal was to create a web-based platform that integrates existing tools, and makes it easier to create new tools in the future. The architectural solution was described as the Tool Portability Profile (TPP), which "provides an environment where tools and the services to support those tools can be dropped in as 'units of expansion' or 'building blocks'". Tools were also likened to plugins. The tools approach allowed the separation of projects around specific functionalities, and it also resulted in the expansion of the system with new tools (such as gradebook, calendar, mail or syllabus[8]).

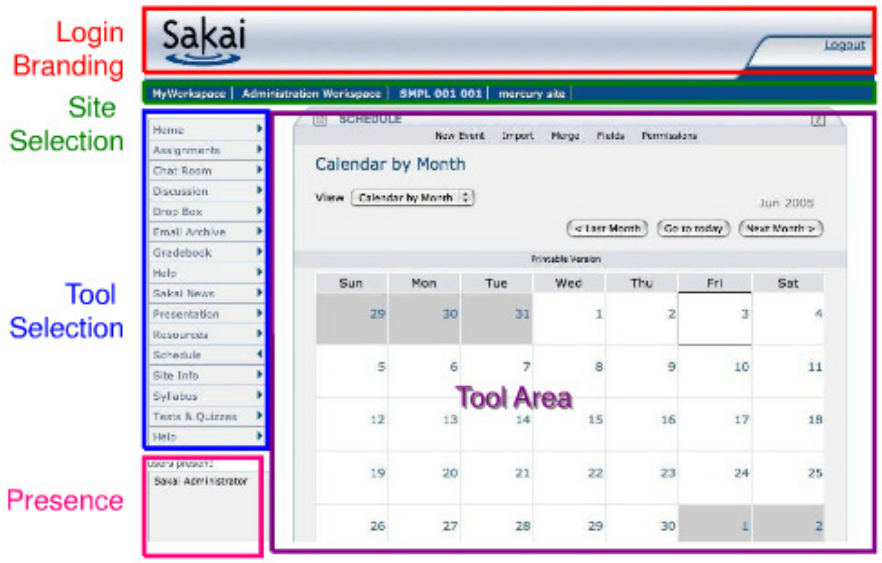
In the original implementation of the Tool Portability Profile, tool services were defined by their interfaces, but the different tools were also allowed to access each other's services. The assignment tool, for example, was able to directly address code inside the calendar tool when a due-date was put on an assignment from the calendar. Thus, the implementation of modularity was only partial.

The tool-based architecture was also apparent in the user interface, underlying the general structure of user experience within Sakai 2. Tools were pulled together in a portal framework, which made them available as elements of the work site associated with each course (see Figure 4.1). In the portal, a menu bar (usually placed at the top of the screen), allowed access to the course site, and within the site, another menu bar (usually placed at

[f] The first Sakai system, Sakai 1.4, which subsequently became Sakai 2, was built from the portfolio system of the Open Source Portfolio Initiative, the testing system of Stanford and Indiana universities, and the CHEF course management and collaboration system of the University of Michigan, which included a variety of collaboration and content tools, like a file manager, a discussion forum and a chat.

the left) allowed access to the specific tools. The tool would appear in the large rectangular area surrounded by the two menus. This architecture meant that only one tool could be used at a time, and it was always embedded within a course site.

Figure 4.1: Tools within Sakai 2's portal architecture[9]



While the tool-based solution was generally considered a good approach for creating entry points to development, the community was not satisfied with the details of the implementation. Ways of improving modularity were investigated at the level of the user interface and the underlying code from the early days of Sakai. Sakai 3 grew out of related explorations.

Toward the end of 2007, developers at the educational technology department at the University of Cambridge (CARET), created a set of widgets for making the information within the learning management system available outside of the university's portal, on Facebook and iGoogle. Related to this effort, they started to explore how they

could create the dashboard-like experience of these popular sites within Sakai 2. This was undertaken as a local development branch (a contrib project), and Cambridge presented the work to the wider Sakai community on Confluence at the beginning of 2008.

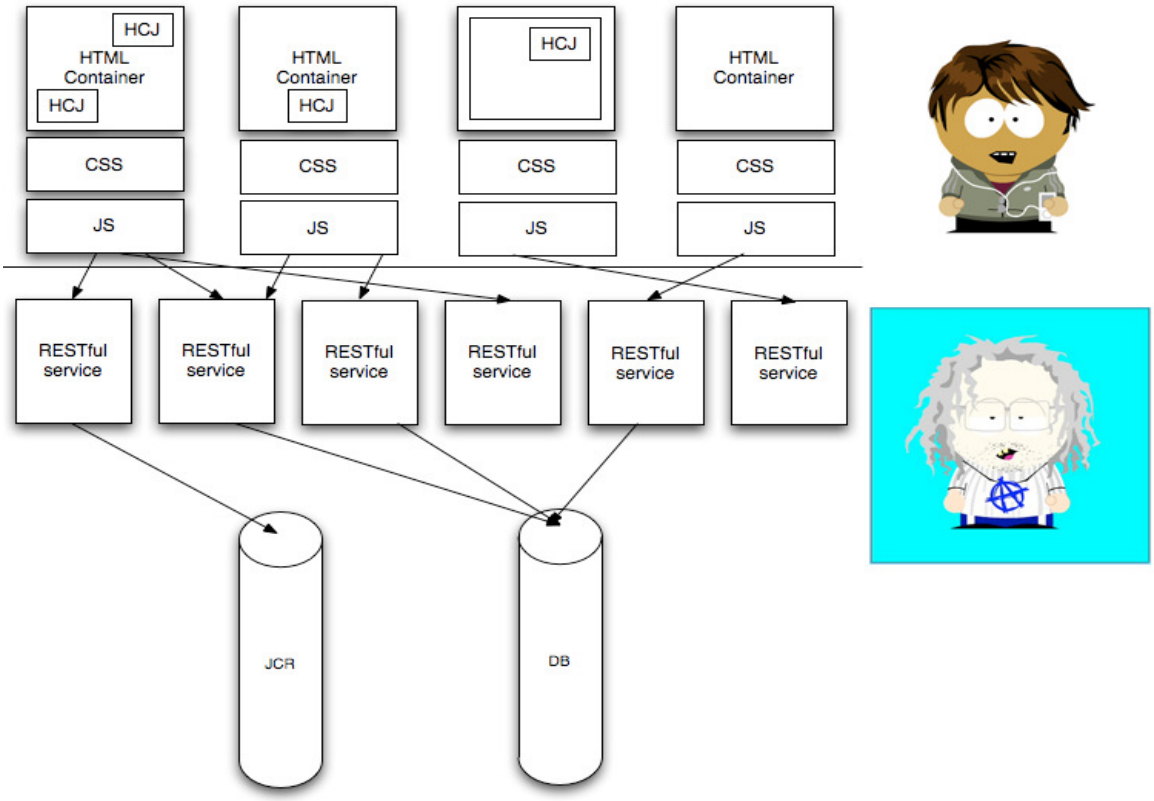
Figure 4.2: The Announcements Widget, relying on Sakai 2 data, is set in parallel with other web-based widgets on a user's desktop[10]

The Confluence space showcased the development as being inspired by the existing example of Facebook's and iGoogle's widget-based interface. In one of the illustrations, reproduced below in Figure 4.2, Sakai's Announcement widget was presented in company with other widgets from the web: a calendar widget, a weather widget and a multi-city clock widget. The widgets presented snippets of information in structured formats, each with its own characteristic presentation logic, and these different

information displays could be aligned on the screen in a dashboard layout. Along these lines, one Sakai-commentator's blog showcased the project as an interesting experiment "mashing up the LMS the Google way."[11]

Figure 4.3: The division of labor between Java and UX programmer in a presentation by Ian Boston about the widget approach.

The Java programmer is portrayed as a socially inaccessible deviant, while the UX programmer appears as a cool guy. The visual argument suggested that UX programmers were better suited to interface with the world.



The Confluence presentation also suggested that widgets were made possible by the separation of back and front end in the software architecture, which allowed user

interface developers to create front end presentations of data without having to understand the complex technical details of data storage and retrieval in the backend (see Figure 4.3). The ease of development with front-end technologies, like HTML, CSS-stylesheets, JSON, Javascript and Ajax was reiterated in several presentations.

CARET made its externally facing widgets available to local students as an experiment in software. The code, the experimental implementation on their local server and the interface screenshots were made available as a proof of concept, which warranted and invited further exploration in the area. A coding workshop was organized at Cambridge to share the ease of development in the proposed approach, with the participation of developers from two other universities.

MySakai tied into a thread of ongoing discussions within the Sakai community about the way content was handled within Sakai 2. These discussions dated back to the early days of the Sakai Community, and the Resources project, which was working on improving the user experience of Sakai 2 content management on the basis of end-user feedback that Sakai participants had brought back from their institutions. Participants in the Resources project looked at various areas of functionality, like adding copyright labels, fine-tuning the release of content to students, and ordering a list of files. During this work, they repeatedly ran into the limitations of Sakai 2's tool-based architecture. Very briefly, this architecture made user experience available in larger bundles, called tools, and made it hard to bring together the bits and pieces of functionality available within these tools. Thus, participants in the project realized that it would be hard to release new lessons to students based on the results of their assignments, because the programmed logic of how assignments worked was locked within the Assignment tool,

and it was not available to the tool dealing with content. They also ran into difficulties pulling existing pieces of content together in meaningful presentations. Thus, it was not possible to create a syllabus which provided links to specific pieces of content within the different Sakai tools, like a document, an assignment or an assessment. Participants in the Resources project started to talk about the need for "breaking up tool silos of content".

From a development perspective these user experience challenges became formulated as the need for "RESTful entities", or making the pieces of content within Sakai 2 tools URL-addressable. To tie back to the concerns of the Resources project, this translates as the possibility of calling up a specific assignment with student-specific meta-data from anywhere in the Sakai 2 portal through a reference, and making sure that the student sees the assignment with correct, student-specific deadlines and submission information. The details of this work are not relevant here, I will only add that the EntityBroker (EB) was successfully created within Sakai 2 for this end, and the development effort paved the way for the subsequent separation of a Sakai 2 back-end (called Kernel 1, or K1).

4.3.3. Creation of a design space for a new Sakai

Aggregating widgets and breaking up tool silos were both formulated as possible pathways for a future Sakai. The first was driven by existing examples, the second by limitations of the existing system encountered in attempting to cater for perceived user needs. CARET even brought them together, floating the "widget approach" as a solution to the "tool silo" problem[g] in informal conversations, but it was the executive director of

[g] "I think what we want to see is a general move towards more RESTful, markup-based strategy for aggregation, in line with John [Norman]'s comment that we would like to

the Sakai Foundation, Michael Korcuska, who was able to combine the two pathways into a vision for a new generation Sakai platform. He did this by pulling the two threads together under the popular umbrella term of web 2.0.

O'Reilly gave an account of web 2.0 as an emergent approach to software arising from the confluence of a heterogeneous assemblage of new software technologies, design patterns, user experiences and business models (2005). Web 2.0 has been associated with a grab-bag of notions like social tagging, crowdsourcing, microcontent, mashup, remix, loosely coupled systems, lightweight programming, scripting languages, and Ajax. These software notions were linked up with wider social values like participation, collaboration or the freeing of creativity, and on this account, web 2.0 may well be seen as a computerization movement telling a tale of emancipation with emerging information technology (Allen, Rosenbaum, Shachaf, 2007; Rosenbaum, 2008).

In referencing web 2.0, Michael Korcuska could be seen as reaching out to a lively computerization movement, but he did not act as one of its promoters (the "opinion leaders" and "pundits" described by Kling & Iacono, 1988; 2001). Instead, he may be understood as relying on web 2.0 to create a *design vision* for the Sakai community, opening up a *design space* by bringing together two previously distinct threads of work within Sakai. The design vision had its first coherent public formulation in a confluence document authored by Korcuska in early 2008, titled as a "Statement of Ambition"[12], and

---

adopt a strategy capable of aggregating "widgets" or whatever from outside Sakai entirely as well as those dispatched internally." In this comment, Antranig.Basman describes CARET's suggestion to create a RESTful approach by relying on widgets.

it was elevated to the status of a formal "Sakai 3 Proposal"[13] by the end of the same year. Korcuska also gave related talks until his final days as the Executive Director of the Foundation in March 2010, and his presentations were later published as recorded podcasts, videos or slides. The following elements were incorporated in the vision:

- A better user experience.
- Multimedia-rich content authoring experience.
- Content organization with tagging and search.
- Social networking for the academic context.
- Academic collaboration.
- Opening up institutional boundaries.
- Service Oriented Architecture.
- Taking advantage of client-side technologies and development approaches.

While there were differences in the presentation, and the line of argumentation became more elaborate over time, the constituent elements were constant across the different formulations. Community members themselves came to talk about "the Sakai 3 vision" as a thing on its own, which transcended these different formulations.

The high-level framing of the web 2.0 computerization movement allowed the creation of the *design space* in two significant ways: first, it provided a perspective from which the endeavor of the MySakai and Resources projects appeared convergent, and second, it lent a sense of urgency and necessity to the suggested line of development.

Related to the convergence of the two projects, first of all it should be noted that the two threads were already partaking of the influence of the software environment labeled as web 2.0. MySakai focused on mashing up content and information along the

lines of the user interface notion of the dashboard with widgets, and explored the underlying software architecture that supported this style of development by separating front end and back end through specific software solutions. Resources focused on local integration of content based on the REST standard of web architecture. All these approaches have been associated with web 2.0. The Statement of Ambition document referred to these projects under the relevant web 2.0 headings, suggesting that they were fragments pointing toward a more encompassing effort, the nature of which had not been charted in all details, but might still be grasped as a unified and cohering whole from the wider angle of apprehension offered by web 2.0 as the integration of diverse content in a single user interface and the related frameworks of software.

The suggested unity was further reinforced by the naming of a future system, Sakai 3, which was prefigured in these fragmentary efforts. The projection of Sakai 3 was based on the existence of code which could be taken as an indication of the feasibility and desirability of the web 2.0 approach. At the same time, no claim was made that Sakai 3 will rely on any particular existing solution. The Statement of Ambition started with the idea that the unity of the future system was based on the "interest" and "express readiness" indicated by the ongoing web 2.0 development efforts within Sakai:

> "This document describes emerging goals and plans for two major releases of Sakai [Sakai 2.6 and Sakai 3], based upon express readiness and interest of several institutions with resources to carry them out over the next year."

Related to the necessity of the new approach to Sakai, Korcuska presented web 2.0 as a domain of existing examples, which had set the expectations and demands for rich and satisfying user experience. On this point, the Statement of Ambition said:

> "Not only have new technologies emerged that allow us to design and build software in different ways, user expectations have changed with the emergence of Web 2.0 technology and the "social" web. We need to take advantage of these technologies and respond to these shifting expectations quickly."

The Sakai 3 Proposal reiterated the same idea:

> "Sakai end users, increasingly familiar with "Web 2.0" technology, are demanding an environment that is more flexible and affords them greater control."

The Sakai 3 vision created a new framing for subsequent design work, and its makeup was significant to this effect. In 2008 the future of Sakai was prefigured in a set of distinct, fragmentary and incomplete software implementations, MySakai, the modularized K1 kernel and the RESTful Sakai 2. These implementations became wrapped within a set of heterogeneous and fragmentary ideas about the nature of web-based systems in higher education. By casting existing efforts in a loose web 2.0 framing, the Sakai 3 vision projected a conceptual space open for interpretation and invited participants to fill in the details by making sense of web 2.0 for higher education and constructing a novel understanding of Sakai as an extension of ongoing development. This is what I suggest to describe as the opening up of a design space.

Because of the makeup of the Sakai 3 vision, the design space was distributed across material artifacts and humans: it was material insofar as it was drawing on existing software, and on the other hand, it was a space of meaning-making and conceptual exploration projected over the existing software. The new design space was also distributed socially. It was embedded in the existing collaborative infrastructure of the

MySakai and Resources projects, and it was inviting wider community participation in building the new Sakai. The web 2.0 framing helped convince a range of contributors beyond the actual software developers that they can and should contribute to shaping the future of the Sakai system. In subsequent years, Sakai 3 received contributions from a growing range of Sakai participants.

In the following chapters, I will describe how the design space prompted epistemic strategies, and how these framed Sakai 3 development. In Chapter 5, I will describe how the openness of the design vision prompted participants to engage in conceptual construction to make sense of the new Sakai. In Chapter 6, I will look at the role of professional practices in this process, and in Chapter 7 I will extend this analysis, and argue that the social perspective was missing from the approaches. Chapter 8 will investigate the epistemic strategies of learning that participants followed to engage with the openness of the design space by pulling in knowledge about the contexts of use. Together, the chapters will provide an account of the DCog processes of conceptual innovation that the formulation of the design space set in motion. Chapter 9 will open up the perspective of the analysis, and link up the local innovation in the design place with the broader context of an evolving landscape of software.

4.3.4. Guiding design through the web of discourse

Online platforms have been described as another significant source of coordination in OSS projects (Gacek & Arief, 2004; Benkler, 2002). An important characteristic of these projects is the geographical dispersion of contributors, which requires that they rely on online communication tools to share their work. Two sets of tools have been discussed as significant for open source. One is code sharing platforms, such as code repositories

(Soundforge, github), versioning systems (CVS, SVN) and related tools for the creation of individualized builds of source code (Maven). These are used are used to organize and coordinate code contributions from a group of participants. The other set covers a wide range of communication tools, which are used to support feedback and discussion related to development. The design of OSS relies on the latter tools, and code sharing tools appear to be limited in use to organizing the code itself. This was the practice found in Sakai, so in the following, I will limit the discussion to the use of communication platforms.

Scacchi suggests that communication tools support an informal requirements definition process characteristic of open source projects (2002; 2009). A functional understanding of new software, or what he calls a set of requirements, emerges from iterative discussions which form a dense web of discourse. The process relies on "software informalisms", such as email and forum discussion, the formulation of scenarios of use or the creation of prototypes. Functionalities will be repeatedly discussed or shared as prototypes with a varying circle of participants, and throughout these repetitions they will become condensed and focused. Functional capabilities may be advanced in technical forms, like screenshots or prototypes, and may be verbally asserted post hoc in discussions.

Scacchi's account provides a good overall description of how the shape of Sakai 3 gradually emerged from several years of discussions. The most striking characteristic of these discussions as witnessed in Sakai is that participants did not attempt to make contributions final, possibly because no one had the authority to do so. Spontaneous discussions of problems might emerge around pressing problems, time and time again,

and then became dropped and forgotten, until the entire discussion came to be revisited as an archive from a slightly different angle. In Chapter 8, I will describe the web of discourse as part of a long-term DCog strategy employed by participants to engage with the contexts of use, and make available domain-knowledge for the purposes of software development.

In this constant web of discourse, I found that Sakai participants actively using three distinct organizing mechanisms to navigate design:

1. Framing
2. Anchoring
3. Staking

Framing involved setting up local contexts of contribution for others to participate in. Projects were created through framing. Sakai 3 as an overarching project came into existence through framing. Just as importantly, framing was used within projects to give direction to activities locally. The most pervasive type of local framing was collection, typically of examples for something. Participants collected:

- existing UX examples, for example for content and group management;
- use cases and scenarios from higher education, for example for content widgets, assignments and group-based activities;
- examples of groups within higher education;
- teaching and learning capabilities.

Framing could rely on two basic components: the creation of an online space and the formulation of the content of the activities. Setting up project-level spaces required Foundation privilege, so a new wiki space in the Confluence, a new issue-tracker space in

the Jira, or a new email list had to be requested. Some projects preferred setting up a Google group instead of a Sakai list, and many Sakai lists were also tracked in external archiving platforms, such as Nabble or Gmane.

Beyond this basic setup, participants could use the communication spaces as they wanted to. Collection efforts often relied on tables. The table with basic explanation could be created in a Confluence page, and advertised in an email. The table could also exist as a spreadsheet in GoogleDocs, in which case the context was created in email, or on Confluence. In both cases, contributors would come and add their own examples. Some collection efforts were successful, others less so. Some had no contribution, or became abandoned mid-way, but they were never closed, and no assessment was made. The collection with the outcomes would reappear if it was subsequently taken up in other efforts.

Confluence was also used to frame the content of projects at the outset, and to keep that framing up while the project was active. Some projects even added a retrospective account after they were finished. Across the web of discourse, project spaces were used as a main entry point to orient newcomers as well as regular participants. The project home page typically included pointers to earlier related work, ongoing efforts and overall goals. The various ongoing efforts both within and outside confluence were often summarized in a page. Because ongoing efforts often did not indicate their status, projects spaces were used for this purpose, but this was informal, like all other activity, and not consistent.

Anchoring involved the revisiting of earlier discussions in support of ongoing work, so as to anchor down and give support to current, tentative work. While many

efforts did not bring fruition at the time of contribution, including the collections I have described, they could be picked up outside of the original context to anchor development. As I have suggested previously, ongoing contributions are not made final in the flow of events, but anchoring allows them to show up as an achievement. Many successful collections became reused this way in subsequent projects.

Staking refers to the act of putting a stake in the ground, and it describes the practice of marking something as an achievement to be revisited in the future. It combines the act of marking for future memory with that of marking as useful and worthy of continuation. The first system created as the new Sakai had to be built with Kernel 1, because Kernel 2 was not ready. Kernel 2 was staked on various occasions to be subsequently included in Sakai 3. For this first system, participants collected a series of educational use-cases around content-based widgets. Eventually, only file upload was implemented, but participants made sure to stake other areas as achievements to carry on toward future phases of work. Staking was an effort to position a contribution such that it should be inevitable in the future, but it meant no guarantee of success. It did not ensure that the work was subsequently revisited: Kernel 2 was included in Sakai 3, but the use cases beyond file upload never became implemented.

Framing, anchoring and staking constituted epistemic efforts at the organization of design. They made rearrangements in online communication platforms to provide focus and finality to the epistemic work of design in a context which could not rely on authority and rules for this purpose.

At the same time, the Sakai Foundation had a complex organizational structure, with an executive director, and with participants who had institutional affiliations.

Korcuska, unlike anyone else in Sakai, had the authority to formulate the vision for Sakai 3 in various documents with special status. Participants would at times also appear as representatives of the needs of their institutions. Both of these authorities could be only exercised within and with respect to the web of discourse. As I have emphasized, Korcuska did not "invent" Sakai 3, he pulled the local threads together under the web 2.0 umbrella. Sakai 3 was deeply rooted in previous and ongoing efforts in the community. Its various elements had been discussed and acknowledged, but not necessarily at the same time, and in the coherent formulation that Korcuska's Sakai 3 vision eventually gave them.

Meanwhile, authority could not be exercised to make things happen, as shown by the institutional authority of participants. Institutional preferences, needs or requirements were sometimes expressed, and staked, but they did not have significant influence on the direction of design before the managed project. Georgia Tech's participant, Clay Fenlason, repeatedly described preferred versions of the system from the very beginning. These expressed needs evolved with the system, and the horizon of possibilities that it was showing at different moments in time. Thus, Georgia Tech started with a preference for assignments, then a preference for a hybrid deployment, where Sakai 3 would provide web 2.0 project sites and Sakai 2 would be used for course sites, then a preference for supporting users contributing widget code, but none of these eventually became a reality in first release of S/OAE. It appears that this release had most to do with NYU's requirements. One reason could be that NYU was the most pressed to use the new system, and volunteered to do a large-scale pilot with the first release. Because they were really pressed to use the new system, they had also drawn up very detailed requirements

with the involvement of different schools, who were ready to pilot Sakai 3. But even these requirements needed the contractual arrangements of the managed project to guide the design of Sakai. Before the managed project, NYU was practically forced to fork its own development to make the requirements happen, because it was not in line with Sakai 3's direction at the time.

4.4. Discussion

In this chapter, I have attempted to provide an account of the Sakai project as a context that has been constructed to support the epistemic work of domain-driven innovation. I have also situated Sakai within the context of open source development, and showed how Sakai has relied on open source strategies to create a distinctly novel organizational approach to the making of software that I have called domain-driven open source.

My goal with the discussion is to formulate a broader theoretical framing for the coming chapters, which emphasizes the agency of participants in creating a cognitively distributed context for innovation. Overall, I am suggesting an account of Sakai as an effort in making a difference in software innovation, which focuses on how software is made in order to make a difference in the product.

My account of the Sakai project has emphasized the work of construction which is involved in making space for innovation in software. Studies of innovation have pointed out the strategic nature of practices related to innovation, notably in connection with standards which have been described as "change agents" that make space for novel technologies in the market by means of coordinating technical design (Bonino & Spring, 1991). A parallel focus on coordination in open source research has resulted in a framing around emergent order in distributed work, and a related tendency to describe open

source practices in terms of social process and recurrent patterning of activities rather than human agency (as suggested for example by the review of Crowston, Wei, Howison, Wiggins, 2012). My case study of the Sakai project suggests that the means of coordination routinely used within open source communities are tools that participants actively and reflectively rely on to construct their own work environment, and to create the conditions of their own work. These construction efforts were not external to and separate from the work that was being done. On every level, endeavors were being constructed as they were already underway: this was the case for the design of the Sakai Foundation, for the new development process, for the managed project, as well as for the various local projects created around Sakai 3. In light of my analysis, the open source movement itself may be understood as an epistemic strategy in support of distributed innovation in software. Open source licensing may be understood as a part of this epistemic strategy that encourages code reuse and fosters the growth of an innovative context of software, from which novel solutions may be adopted.

I have also shown how the design space created around Sakai 3 was growing out of the broader context of open source practices in Sakai, and argued that an account of innovative practices within the Sakai 3 design space requires that we take into account its cognitive character. Most importantly, I have suggested that the Sakai 3 vision projected an open conceptual space as an extension of ongoing development, and invited interpretative engagement with tentative artifacts to make sense of web 2.0 for higher education. The following chapters will be gradually completing this argument, as I will show how the design space resulted in a novel solution, and discuss various aspects of the knowledge that was informing this solution.

In the discussion below, I will argue that a cognitive and distributed framework is necessary to account for innovative practices. I will further suggest that the framework implies the outlines of an account of human agency related to "how humans create their cognitive powers by creating the environments in which they exercise those powers", as Hutchins suggested (1995a, p. xvi).

Relying on the framework of distributed cognition, Nersessian (2012) has shown the importance of reflexive epistemic practices whereby scientists build environments for thinking which make their epistemic work possible. Her case studies describe how two engineering labs built surrogate artifacts to serve as models of phenomena that could not be directly observed and controlled for the purpose of experimentation. Going further, Nersessian has also shown how the labs were initially created in connection with the models, suggesting that the models were designed by future lab leaders to afford collective conceptual construction, and in this manner, they were constitutive of the evolving distributed cognitive spaces of the labs. They represented a promise and potential for future findings – one of the experimental devices was actually described by participants as a "big gamble" –, and in doing this, they were constitutive of a loose trajectory of conceptual construction.

In my analysis I have argued that the Sakai project, and more centrally the Sakai 3 efforts were similarly built as distributed cognitive spaces to support collective conceptual innovation in software. I described Sakai 3 as a design space with an open-ended directional horizon, which was to be filled in through interpretative engagement with tentative artifact. The coupling of artifacts with conceptual constructs was central in both cases: their particular affordances made epistemic work possible, and in this way

they were instrumental in making space for conceptual growth. Nersessian, Kurz-Milcke, Newstetter and Davies (2003) called these spaces evolving distributed cognitive systems. I have coined the term design space in supplement to this, to focus on the constitutive gesture of making space for design, which resulted in the subsequent process of conceptual evolution.

It may be added that the contexts of science and technology development emphasize different goals, which are mirrored in the coupling of cognitive and material elements, and result in constellations that are characteristic of the two epistemic fields. Science represents a constellation in the coupling of cognitive and material elements, which is geared toward building conceptual structures (theories) that are responsive to real world constraints. A central strategy for making theory in the sciences is the construction of experiments, where conceptual structures are modeled as a dynamic process, which can be run under various constraints for the purpose of exploring the implications of the structural setup. Experimentation may be understood as a distributed cognitive process, where conceptual structures become coupled with artifacts in the material world. Setting up experiments in this distributed cognitive sense is subject to practical constraints, which means that experiments emerge as conceptually driven approximations of real world phenomena. In the research cited previously, Nersessian and her colleagues (Nersessian et al., 2003) have described how researchers built in vitro physical models of in vivo biological phenomena which would be hard or impossible to study directly within a living organism. Technology development inhabits a distinct constellation, where the relationship between conceptual and material elements is different: the purpose is to build artifacts which are operational within real world

constraints, and conceptually-driven approximations analogous to experimental model devices are used to assess what may work without having to build it in full.

In spite of these differences, it appears that design spaces are central for understanding the work of innovation that is done in these settings. Design spaces are created in view of supporting collaborative epistemic work, and they are constructed with an open-ended directional horizon which affords conceptual innovation and epistemic growth. In the following I will show that the framework of distributed cognition is necessary for providing an account of how design spaces work.

First and foremost, the design space centrally relies on a conceptual setup that invite the application of human cognitive powers to construct new understandings, while also providing means for engaging in conceptual construction.

Second, the design space represents conceptual arrangements in preparation for future epistemic work, and because of this, temporal framing is central to its cognitive setup. It may be argued that major human cognitive capabilities arise in connection with a temporal framing. Loose future-oriented conceptual structures like those associated with the design space appear to be present in plans for the future, as Miller, Galanter and Pribram's (1986) or Suchman's (1987) account suggest. Vygotsky (1978) also describes the higher mental function of intentional memory as the "construction of a process of memorizing":

> "When a human ties a knot in her handkerchief as a reminder, she is, in essence, constructing the process of memorizing by forcing an external object to remind her of something; she transforms remembering into an external activity. […] It may be said that the basic characteristic of human behavior in general is that

> humans personally influence their relations with the environment and through that environment personally change their behavior, subjugating it to their control. It has been remarked that the very essence of civilization consists of purposely building monuments so as not to forget." (p. 51)

The above passage is commonly cited to underline the outside- in regulation of human thought with the help of external signs. My point here is that Vygotsky describes the result of external regulation as a process, memorizing rather than memory, with a temporally constructed character.

The design space appears to represent a particular engagement with the future, which is characteristic of innovation in that it projects loose conceptual structures to be filled in by cognitive construction. The temporal framing places the design space within time, and requires that participants engage with time as they navigate its conceptual structures. In my analysis I have suggested that participants in the Sakai 3 project thought of their work as situated within a temporal flow, and referenced their actions in connection with this, placing stakes for the future, anchoring themselves in the past, and making preparations for future work by gestures of framing.

Thirdly, artifacts are central for the cognitive and temporal arrangements that design spaces accomplish. With respect to this, I want to emphasize the coherence that they endow on conceptual construction over time. Nersessian (2012; Nersessian et al., 2003) highlighted how experimental devices were built to achieve research goals, and how their history of construction provided rationales of ongoing work. In the context of technology development, the projection of a future system created a loose but high level

indication of coherence that participants were seeking to reconstitute through their conceptual construction, as I will argue in more detail in the subsequent chapters.

Finally, the design spaces I have discussed were created with the aim to make possible contributions by others. For this, they had to be "grandiose" in terms of the scale of both their conceptual promises and affordances, to invite and accommodate wide-scale participation. In connection with Sakai 3 I have argued that the creation of the design space was important for inviting further participation in a context where work was already predicated on some form of local collaboration. The particular construction of the space was also important: the web 2.0 framing, and the related focus on user experience, created a conceptual space that accommodated a broader spectrum of collaborations, notably from non-developers, and also outlined a division of labor where knowledge about the user domain was positioned to lead the development process.

In the above I have argued that distributed cognition is necessary to account for the epistemic construction involved in processes of innovation in science and technology development. I will now show the implications of a distributed cognition account for making a difference in innovation, which can be understood as a cognitively-based account of agency.

Agency has been a central and problematic concept in social theory, notably in connection with the influence of the social. The problem was generally framed with respect to the constraining powers of a social environment, and how much latitude or freedom this may leave for human action. My goal here is not to enter the long and complex history of this debate, but to suggest an outline of how we may conceptualize human agency for the purpose of an account of making a difference in innovation.

Social constructionist accounts of innovation in technology and science argued against technological determinism by suggesting that the content of technologies was the result of non-technical selections made by relevant social groups (Bijker, 1997). Bijker and Law (1992) argued that the processes of social construction of technology implied contingency in the evolution of technologies. This argument focused on the significance of a broader social environment in shaping innovation, and downplayed the role of agency in shaping technologies. Knorr-Cetina's work (1993; 1984; 1981) has been looking at the socio-material situatedness of epistemic practices in the sciences. She investigated the occasioned and occasioning character of scientific practices, or how scientific work makes selections which create its own environment, and how scientists subsequently rely on this environment to validate their work. Related to this, Knorr-Cetina emphasized the complex, socially and materially distributed machinery of science at the expense of agency, notably in her account of the technological mega-project of the Large Hadron Collider (1993). She also emphasized the ad hoc nature of scientific work, which relies on what is at hand for building experiments and arguments (1981). While both Knorr-Cetina and SCOT talk about selections, and make an argument that is based on the directionality that selections imply, they set technical deterministic accounts in contrast with the contingency of the social. Because of this, they do not attempt to grasp the directionality of socio-material practices, or how human contribution may shape that direction. In contrast, the sociology of expectations emphasizes the trajectory that results from selections in technological innovation, but appears to suggest that the process of selection is primarily social, and it is neutral for the content of innovation (Van Lente, 1993, Belt & Rip, 1987).

Looking at epistemic work with the lens of distributed cognition as a sense-making project of construction allows us to outline an understanding of human agency in the shaping of epistemic objects. Most importantly, the cognitive account makes it possible to conceptualize the content of the epistemic process as that which becomes implicated in transformations by socially, materially and temporally distributed human cognitive powers. In social accounts, the analogous understanding of content appears as interpretations or representations, and this content becomes implicated in selections of a social nature. Second, distributed cognition allows us to see how a social and material environment becomes implicated in the distributed cognitive process as people "create their cognitive powers by creating the environments in which they exercise those powers." (Hutchins, 1995a: xvi)

In the above, I have described how a design space was created from a local environment as a cognitively empowering selection which creates new cognitive powers by providing a loose direction to epistemic construction. The Sakai Foundation was designed as a space of distributed collaboration of knowledge workers from higher education to bring about domain-driven software development, and within this broader context, the Sakai 3 design space was created to allow for domain-driven innovation. I have argued that the cognitive-epistemic character of the design space was central to its role in fostering innovation in software. The account of the Sakai case is paralleled by Nersessian's analysis of the evolving distributed system of engineering labs (Nersessian et al., 2003).

The theoretical apparatus of distributed cognition allows us to account for the direction of innovation in terms of conceptual trajectory, and to investigate how a

trajectory has been made possible by selections on knowledge. This is the sense in which I would like to talk about agency, or the power of humans to make a difference in their world. Human agency gives shape to new technology in a distributed cognitive process, which is not deterministic, but endows change with direction, as suggested by the notion of the design space. With this, I do not mean to question Knorr-Cetina (1993) and others that epistemic work is situated within a social and material environment that is not of its own making, and its efforts to fend the odds of the environment may be crossed by powers more powerful. Their accounts remind us that agency should not be understood as a deterministic control; chance and indeterminacy need to be accommodated by the notion. At the same time, distributed cognitive human agency is at work to create controls over the environment in view of making space for particular kinds of epistemic work, and it is also already relying on the securities made available by histories of distributed cognitive achievements in culture. This shows agency as situated within time.

With the above in mind, the account of design space shows agency at work in creating innovation within science and technology development in a distributed cognitive bootstrapping process, where an open cognitive-epistemic horizon is first created to be filled by conceptual construction. I suggest that the directionality of this process may be understood with the help of the concept of bounded indeterminacy, which I have borrowed from developmental cultural psychology (Valsiner, 1987). Bounded indeterminacy is a systems concept, which was created to describe how human capabilities, and notably cognitive capabilities, outline a developmental process which has directionality as well as an open horizon, wherein individual and cultural diversity

can grow. Agency at work in design spaces may be seen as creating a space of bounded indeterminacy and subsequently filling it with directional growth.

In this first chapter, I have shown how the Sakai 3 design space was created. In the remainder of my dissertation, I will look at how epistemic practices facilitated within this space led to the design outcome. I introduced the notion of agency above to emphasize that the design outcome was made to become the way it was. Participants were collectively making particular epistemic selections, and these selections could have been different. In the following chapters, I will look at significant selections related to knowledge, and show how they were implicated in the resulting design outcome. In Chapter 5 I will describe a case of conceptual construction within the design space, and show how the space was generative of a conceptually novel outcome. In Chapter 6 I will show that participants of the design space were relying on epistemic tools from the professional context to effectively create a user experience orientation to design work. Chapter 7 will point out a missing conceptual perspective in the design process of Sakai 3 in connection with an account of the failure of the system, and outline an alternative approach to design which goes beyond user interaction. Chapter 8 will show how the conceptual process was reaching out for knowledge about the context of use, and how the user experience orientation commanded the orientation towards individual users' interaction with web-based software. Chapter 9 will look at the role of a wider context of open source software in the creation of the design space.

CHAPTER 5. CONSTRUCTING A NEW CONCEPTUAL MODEL OF USER EXPERIENCE

In the previous chapter I described how participants were making space for a new system by connecting various threads of work around Sakai 2, and how a design space was created by the formulation of the Sakai 3 vision. In this chapter, I will rely on the theoretical framework of conceptual modeling to show how the openness of the design space led to the local innovation of a new area of functionality in the platform.

Research has pointed out the importance of a possibilizing element in design and innovation which makes way for the novelty that results from these processes. In science and technology studies, the term expectation has been used to refer to broad descriptions of new possibilities in technology which guide the process of innovation (Borup et al., 2006). Expectations have been described as future-oriented abstractions about the promise and potential in new technological directions, and related research has emphasized their generative contribution, which anticipates new technology without defining it entirely. In his case study of the emerging membrane technology, Van Lente suggests for example that participants had no good answers to the question of what was a membrane, nor did they know how to go about creating one, but they were convinced that membrane technology had potential (Van Lente, 1993; Van Lente & Rip, 1998b). Van Lente and Rip (1998a) have also argued that expectations are generative of a protective space or niche, which guards from potential challenges to uncertainty while the details of new technologies are worked out. In this process, the shape of new technology is defined as promises become converted into requirements. Researchers in the area clearly

acknowledge the cognitive aspect of expectations (Van Lente, 1993), but their account of how they generate innovation is restricted to social processes, and they do not attempt to describe the epistemic processes whereby uncertain abstractions become requirements and new technology.

Folkmann (2013) argues in the phenomenological tradition that aesthetic design (or shortly design, as in art and design) has a possibilizing element, insofar as it is able to project an abstract idea of possible future forms and functions. He describes possibility as situated at the threshold between actual and new:

> "Possibility evolves at the threshold of actuality. It is a dimension of the actual that both transcends the actual and is inherent in it. Through this double constitution, possibility can push the actual in new directions (and hence is bound to it) or break free new dimensions of meaning (and hence transcend it)." (Folkmann, 2013: 20)

In sum, Folkmann, Van Lente, Rip, and other researchers from the field of the sociology of expectations have convincingly argued that there is a cognitive-epistemic possibilizing element in design, which connects the actual with the new, but have not accounted for the process in cognitive-epistemic terms.

In this chapter, I will present a case study of conceptual modeling that attempts to throw light on this process. The cognitive basis of conceptual modeling is mental modeling which involves the construction and manipulation of malleable cognitive structures referred to as mental models. Mental models have been characterized as structural relations across representations of knowledge, based on:

> "the general hypothesis [within related cognitive research] that some mental representations of domain knowledge are organized in units containing knowledge of spatio-temporal structure, causal connection and other relational structures" (Nersessian, 2008b: 398).

Research looking at conceptual growth in the sciences has shown that the new conceptual structures of scientific theories do not result from sudden bursts of inspiration. Instead, they are the result of an often slow and circuitous process of construction, which involves exploring the possibility of making new connections within the constraints of existing conceptualizations and real world phenomena. Conceptual modeling is a process distributed over time, and Nersessian's account has emphasized the role of model representations that make conceptual modeling distributed (2008a). Her ethnographic study of research labs has also shown that the process becomes socially distributed across various participants, and social processes of collaboration unfold in connection with model devices (Nersessian et al., 2003).

My goal is to show that the open-ended character of the Sakai 3 design space embarked participants in a sense-making process, in which they were constructing conceptual models of a possible design by using thought experiments ("what if" scenarios) to explore connections between elements of the design space. The result of this process was a novel user interface, using the familiar dashboard framework to support group-centered user experience.

5.1. Designing a new interface: conceptual construction from content widget to group dashboard

5.1.1. The starting point: the content model

In the original discussions that gave rise to the Sakai 3 vision, groups did not have a central role to play. The design task was framed around the reorganization of content. The Resources project formulated a dissatisfaction with Sakai 2 architecture in terms of content being trapped within tools. In Sakai 2, content could be accessed in the context of specific tools. When an assignment description was made available in the assignment tool as a text or a downloadable file, it would not become available in any other tools. If the same assignment had to be discussed with the help of the Forum tool before submission, it had to be copied or uploaded there. "Breaking out of tool silos for content" was a phrase repeatedly used in community discussions and in Michael Korcuska's presentations to describe the need to go beyond this state of affairs. This conceptual model contained the elements of tool functionality, content, context of presentation and storage. The problem was identified as the coupling of the context of presentation for content with functionality and storage.

In attempting to mash-up content functionalities within the shared context of the page, the MySakai development was equally operating with a conceptual model based on the connection between content, functionality and context. The widget model stated that the widget can combine any content with any functionality, thus serving a range of contexts. It also suggested that underlying the user interface should be a generic content store, with a structure that does not mirror that of the user interface. I will call this conceptual model the widget-model of Sakai 3.

Figure 5.1: My diagram of the widget model as a context of functionality and content

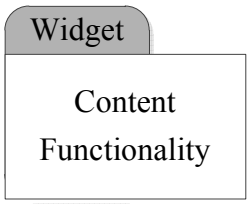
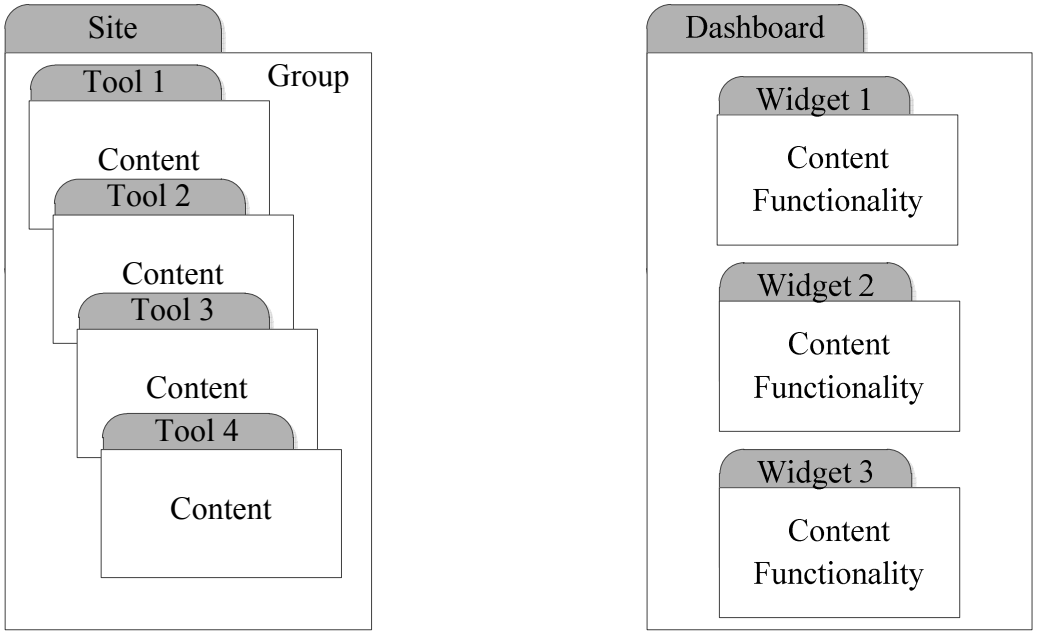


Figure 5.2: A comparison of Sakai 2's course site with a dashboard-type page, based on my diagrams



The Resources project encountered the content-problem in connection with its attempts to create a better structuring of content within Sakai 2, and the quest for solutions became framed in the content structuring paradigm. At the time when the Sakai 3 vision was being formulated, individuals started to contribute synoptic conceptual

overviews around content organization. The Content authoring project was created to house these spontaneous solution attempts in a collaborative context.

The contributions included:

- A hierarchy of three layers of contexts for discussing authoring (academic programme, course, teaching unit).[14]
- A UML-diagram of the Portfolio tool's model of content.[15]
- An authoring taxonomy.[16]
- A chart representing types of content in terms of granularity and degree of structure (Figure 5.3),[17] and an attempt to approach the same problem in a diagram (Figure 5.4).[18]
- A technical discussion of the software plugin architecture.[19]

Figure 5.3: A chart with a conceptual model that accounts for content types in terms of granularity and internal structure[20]

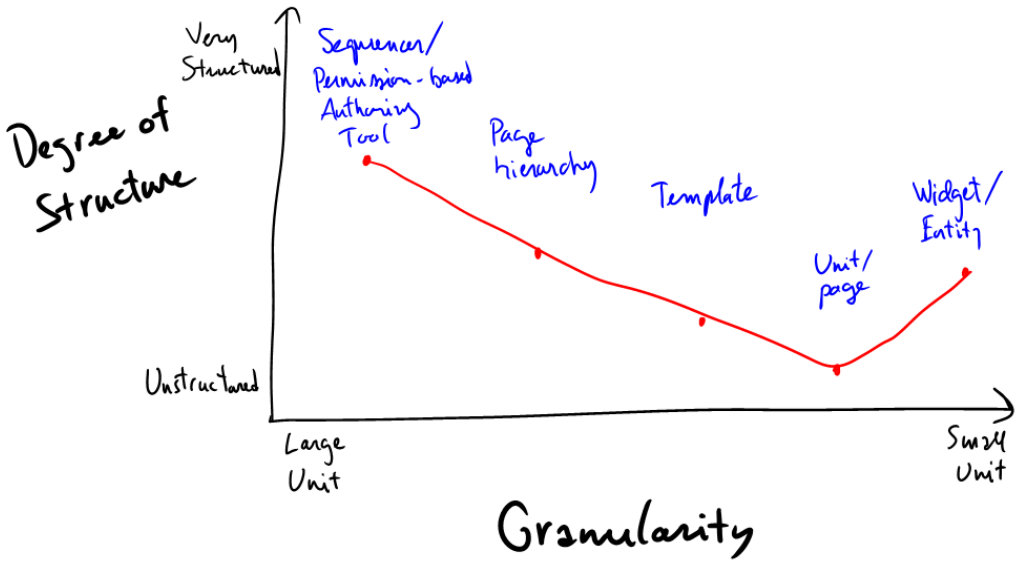
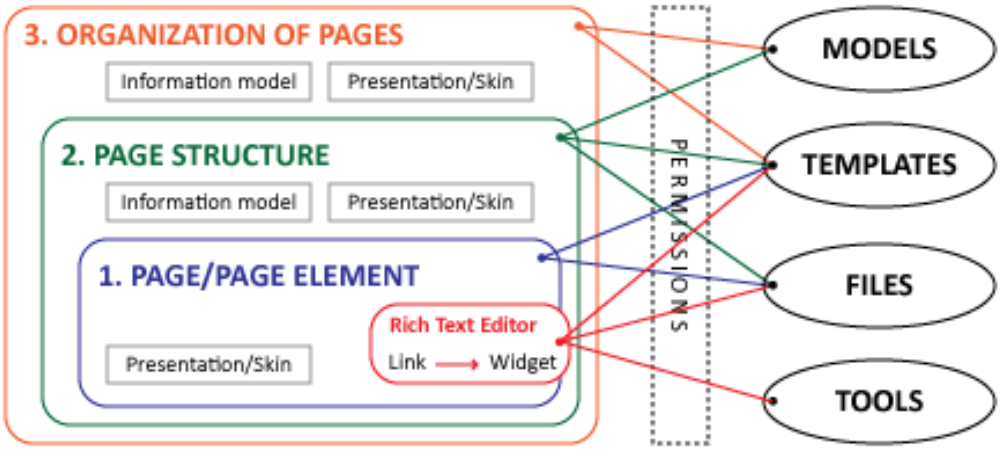


Figure 5.4: A diagram of the conceptual model for hierarchical levels of authoring[21]



The various contributions showed the extent to which the design space was radically undefined, and indicated the need to master the conceptual space related to content. At the same time, they limited their exploration to content structures: What should be the units of content to display? How do those map onto existing units, like files and web pages? What structures are created from these units? What are tools to manipulate content? Are there existing technologies to borrow for this? The conceptual models attempted to suggest candidate answers by creating convincing synoptic models where things came together, as exemplified in the following suggestion (I have italicized passages that mark uncertainty):

> "Discussions … *suggest* that we are trying to build a tool that allows pages (web pages) to be created and linked together to form *larger structures (currently undefined*). *At a minimum*, we need to provide a way to author content marked up using HTML, *likely* using something like TinyMCE or FCKEdit. I would like to propose that we consider a "page" to have structure over and above that provided

by HTML that is made up of one or more "elements". [After this comes the discussion of the suggested plugin architecture]."[22]

As strides were being made in the widget-based redesign of the Sakai 2 home page, the widget approach emerged as a promising direction for structuring content. Nathan Pearson was tasked to continue the widget-based redesign with course sites, and the 3akai project defined its first goal as the formulation of what widgets may be for content in the higher education context.

Meanwhile, Sakai 2 had other inconvenient couplings, which came to the focus of attention in the process of widget redesign. In Sakai 2, course site was a central organizing element, serving as a workspace which defined a context around functionality and people. People could access content because they were members of a site. Because Sakai 2 made tools available within course sites, content was doubly trapped when it came to sharing. This was cumbersome for research projects and other forms of academic collaboration, where documents would be regularly shared among a group of people, but no complex functionality or workspace was needed. This also made things challenging for teams within courses, who would have needed a workspace with the privacy of shared content, but could not have one. To complicate things further, users' access to functionality was to some extent defined in terms of their roles within a course site, and the set of available roles was also hardwired (with such roles as student, instructor, teaching assistant or guest).

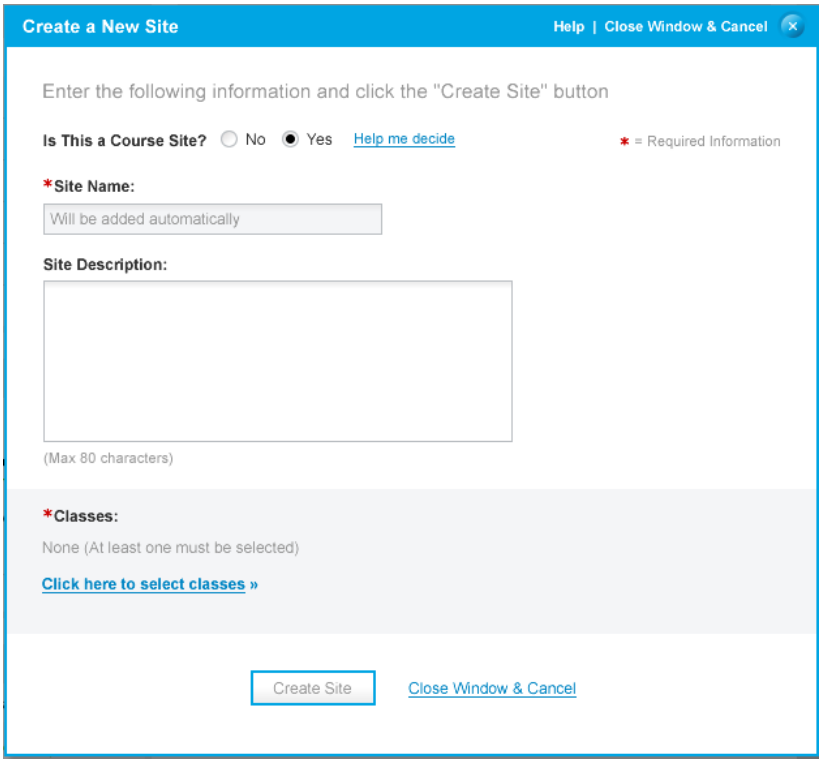
5.1.2. Groups enter the scene

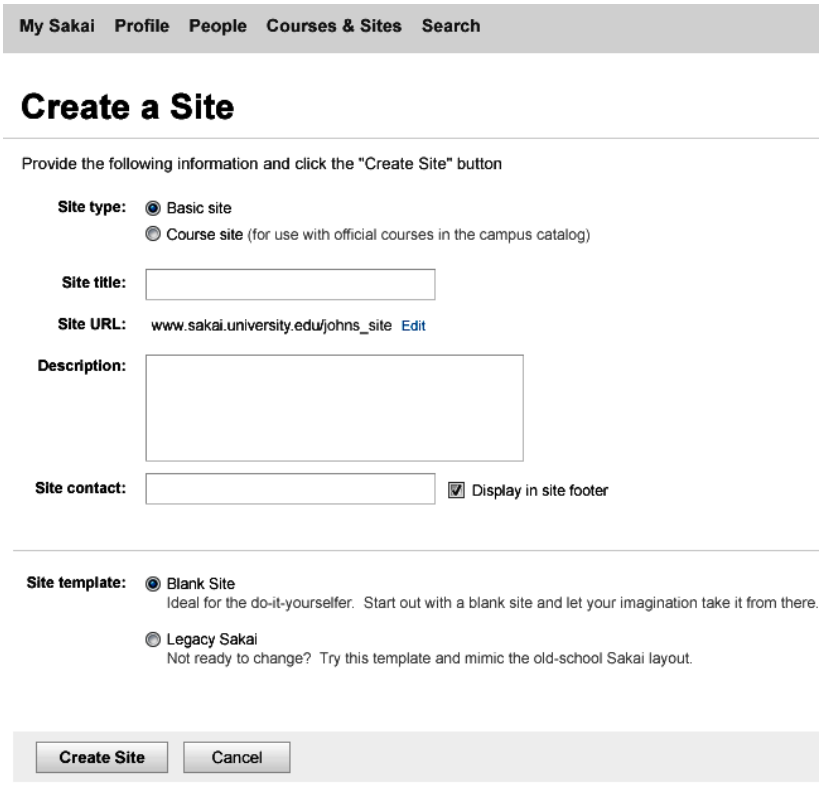
Groups first became a focus of discussion when Nathan Pearson set out to design the dashboard layout of course sites and the attendant management functionality. He

documented what he had learnt from Sakai participants about the course creation process as a page of requirements, and created mockups on the basis of these requirements. In the Sakai 2 world, the workflow of creating a course site associated a group of students (a class) with a site, and this was the approach appearing in the requirements and mockups:

> "1. Creating a single course site per class that he teaches, one at a time:
>
> The instructor will be presented with a site creation process to build a site based on an official university record of a class he is teaching. This will involve finding a record of the class he is teaching via some type of navigation process (ex: browsing and/or searching). Likely, he will need to find his class by filtering options which may include academic year, course subject, etc.
>
> Once he locates his class in the system, he can build a course site for it. […] If the class is not in the system, he can submit a request to create a course site for a class that he thinks should be in the system."[23]

Figure 5.5: Original and redesigned site creation wireframes. The first frame requires the association of a class with the site, the new one does not.





The mockups received a number of critical comments suggesting that site creation should not be entangled with site membership. As one participant summarized this point:

> "Speaking more generally, this process of mapping the registrar's enrollment units to sites seems fundamentally a "membership" issue, and not a site creation issue, so that linking sites with users inextricably (as it seems they are here - you're forced to do this as a necessary part of course creation) might not be the most helpful way to structure the tasks."[24]

The point was further supported by two use cases, which showed that an instructor might want to create a site before the registrar had made available the class roster to be associated with it. Other miscellaneous details were provided about the messiness of adding non-student members, like teaching assistants, to an existing course manually.

The episode resulted in Nathan Pearson's admission of a lack of understanding of basic notions in Sakai, and a desperate attempt to involve the community in making sense of things:

> "After wasting a considerable amount of time sketching thumbnails and wire-frames, I've come to realize that any design I come up with for course management just doesn't hold water! […] I think the problem stems from trying to beautify an existing design that I don't fully understand.
>
> Some of the issues that confuse me are:
>
> Are site types [the course site and the project site] something truly special, or are all sites essentially the same with the exception of different access controls?
>
> Is adding rosters during the course site creation process a required logical step or just a legacy design decision that we've all grown attached to?

> What the heck are rosters? I mean, I think I know, but is that word common end-user vernacular?
>
> […]
>
> And since rosters represent only one of several ways of adding users to a site, do they deserve the special treatment they get in the UI, or should adding site members be a more comprehensive experience with rosters, campus directories, and external users all managed through a related set of screens?
>
> Please help shed light on these and other points I may not have thought of by using the comments feature on this page."[25]

The first item talked to Sakai 2's distinction between course site and project site. The term 'project site' described cases that were unlike a 'course site' in that access was not based on institutionally defined course membership. The question was asking whether the conceptual model based on access control was sufficient to make sense of the difference between course and project sites, or a more complex conceptual model was needed. A possible response to this question could have been that project sites did not need roles or a workspace, but course sites did – in this case, project sites and course sites could be distinguished on the basis of the additional conceptual complexity that roles and workspace brought with them. The second item expanded the previous problem from the perspective of the course site, asking whether the connection between class membership and site was conceptually constitutive of a course site. The last items addressed the status of course membership lists, called rosters in Sakai 2. It only pointed in a direction that later became the topic of lengthy discussions: what are lists of people (like a roster), how

are they different from groups with a purpose (like a course) and should they have full status in new Sakai?

At the turn of 2008 and 2009, participants around 3akai found themselves engaging in discussions about similar questions related to the status of groups in a web 2.0 conceptual model of content in Sakai 3. In the following e-mail excerpt, Clay Fenlason summarized one such discussion in an online conference:

> "[A] central theme of today's 3akai design discussion was grouping. I thought I'd try to recount what seemed to me the key points, partly as a record for anyone interested, and partly as an exercise to hone my own understanding of the ongoing conversation.
>
> The issue in a nutshell is how loosely or tightly coupled the user experience of groups and sites should be.
>
> There was an argument made (and much belabored by a windy sort) for fairly loose coupling. That the group is fundamentally a collection of people, and the more tightly we bind this notion to spatial concepts (like "site" and "page") the more we weigh it down with baggage that can make it problematic to interact with groups in lightweight ways (e.g. share this folder with everyone in *this* group; post this announcement for everyone in *that* group; send this message to these 3 people and those two groups). Conflating groups and sites muddies the waters. Added to which, there seem to be plenty of cases where one-to-many and even many-to-many relationships between sites and groups would be advantageous.

> At the same time, no one disputed the notion that most groups would want *some* site to be part of, and many groups would want their *own* site for their particular groupiness. […]"[26]

The summary contains a suggestion for the combination of two approaches for connecting groups and content:

1. In the underlying model, groups and sites should not be connected at all, to allow for maximum flexibility for how they become connected (no connection with "lightweight" groups which have no site, one-to-many and many-to-many relationships).
2. The user experience model should support the different connections in a meaningful way. Groups that want a site of their own should be able to have it, for example.

After the initial discussions about groups, Michael Korcuska attempted to provide an account of groups in the context of collaboration.[27] The account was based on the following conceptual elements:

- A group being a collection of people and other groups.
- A space being a collection of content and functionality. (Note the parallel with the conceptual model of widgets.)
- A role being a collection of permissions over content and functionality – "what a user can do in a given context."[28]

He pointed out that in Sakai 2's model, roles were associated with groups (i.e. a course group would assign student, instructor or teaching assistant roles as provided by the course). He suggested to tweak this model for the purpose of Sakai 3 by associating roles with spaces.

"Most things that a user can do in the system will happen in a space. For that reason most roles and permissions should adhere to the space context, and not the group. This also allows for people to have different roles in different collaboration spaces despite a common group. Who you are in a group should not necessarily determine what role you take on within a given collaboration context." [29]

This approach turned space into a bridge between collections of things and collections of people:

"a convenient way to group together content and functionality that you want a group(s) of individuals to have similar access to." [30]

Figure 5.6: My diagram for Korcuska's suggested collaborative model of a site

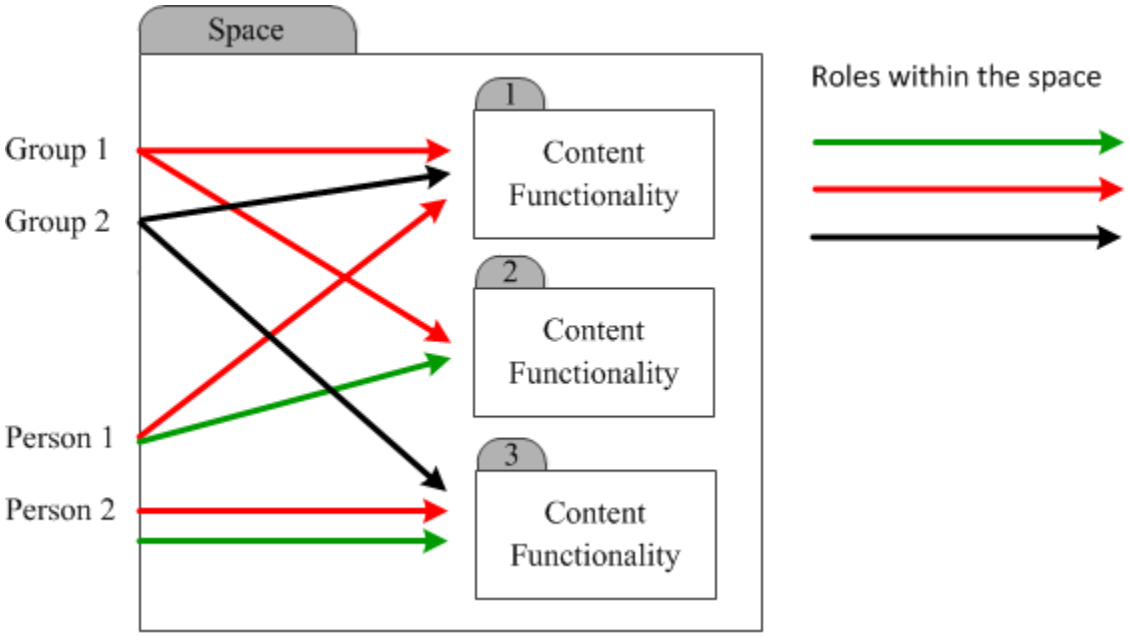


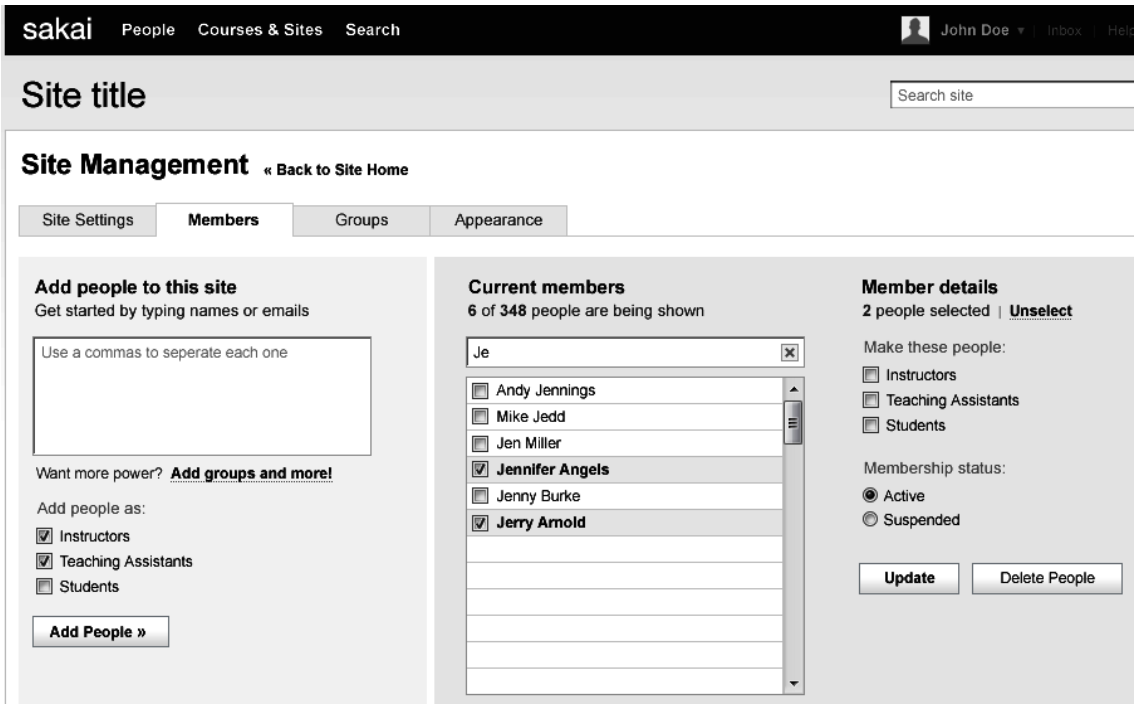
Figure 5.6 shows a possible diagrammatic rendering of Korcuska's account of groups. The account may be seen as a derivation of the widget-model inherited from the

Resources and MySakai projects. Instead of being simply a context with content and functionality, a site is now a collaborative context which contains content, functionality and roles. Persons and groups are connected to content and functionality by roles, but they exist outside of the site. Note that Korcuska's account is ambiguous in several respects:

1. It does not say whether the site is a collection of different kinds of contents and different kinds of functionalities, or, as the diagram suggests with reference to the widget model, a collection of bundles of content and functionality.

2. It does not specify whether roles are specific to content and functionality, or generic, as my diagram suggests.

Figure 5.7: Final design simplified to adding individual members to a site[31]



Soon after these initial explorations, the implementation of group-related features was set aside, and group management was also dropped from the agenda of 3akai. Group-related functionality was systematically removed from the mockups. Figure 5.7 shows a design, where people could be added to a site, and there was only a dummy reference to adding groups). The narrowing of 3akai's scope in terms of groups was accounted for as a temporary focus on project sites. For the purpose of 3akai development, project sites were understood as a simple case focused on content and collaboration, in comparison to course sites, which also involved the complex group and role relationships characteristic of education.[32] The concept of project site as the alternative to the course site had emerged in the Sakai 2 context. At some institutions, project sites were made available in Sakai 2 as a type of site different from course sites, catering for all kinds of non-course-related collaboration in the institution.

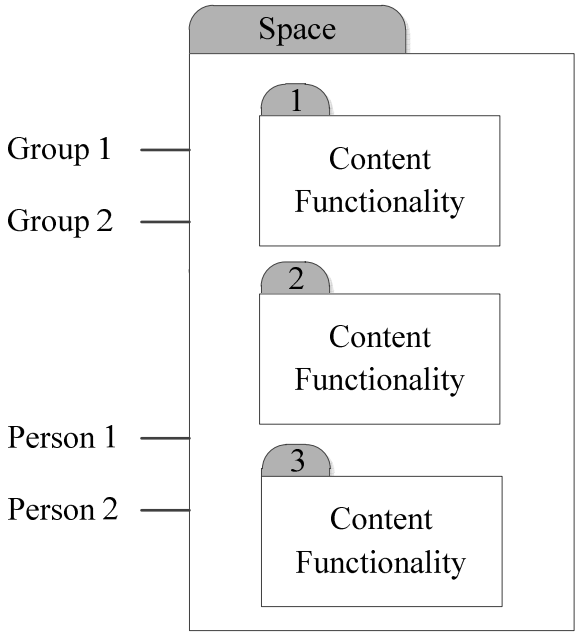
The focus on project sites as lightweight versions of collaborative spaces contradicted some of the design insights that had appeared in the project. Most importantly, it contradicted the idea that content could be shared with a list of people without the requirement of a shared space. Project sites were only different from course sites in their name, otherwise they inherited the structural setup of course sites in Sakai 2. This specifically implied a strong connection between site and group. As one participant retrospectively described this orientation:

> "The current "open experiment" of 3akai and K2 was driven by Nathan [Pearson]'s earlier UX proposal, which in turn was heavily influenced by Sakai 2: we see social networking, content management, and dashboard features added to a friendlier version of "Worksite Setup". Because of this, "Site memberships" are

the only groups the current 3akai demo can create, browse, add to and remove from."[33]

Figure 5.8 presents the implementation as a derivation of Korcuska's model, from which roles have been removed. The possibility of having groups beyond sites was kept open, but at the level of the actual implementation, groups remained strongly coupled with sites, just like in Sakai 2.

Figure 5.8: My diagram for the implementation of the collaborative model



5.1.3. Exploring new conceptual models of groups

Meanwhile, in informal avenues participants were repeatedly sharing their speculations about what would happen if things were looked at differently. In the summer of 2009, the Groups project was created to explore the same issues in a more organized manner. The

suggestions that were explored in these thought experiments and design explorations can be grouped around four general insights:

1. Groups could be showcased by Sakai 3 from the perspective of group identity. Groups could have a profile page like people, so that they are discoverable.

2. What if groups were also primary constituents, like sites, being joinable, discoverable and having their own tools and content?

3. There is a conflation between groups and sites. What if we simply thought of groups as lists?

I will illustrate each of these with a small number of examples before going deeper into the analysis.

1. There could be a special social networking type of group, with a profile page like people, so that they are discoverable.

The suggestions in this group were exploring the idea that groups could be treated like persons in the user interface: they would have a profile, which is informative for what they do, and allows for others to communicate with all members of the group.

1.1.

> "I'm not certain how best to account for this, but I'm starting to wonder if or how social-networking-type-groups would be treated differently. They would tend to be focused more on group identity and member communication than collaboration, for what that's worth. Perhaps they would only have a profile by default. Would this mean they wouldn't tend to have a space, although they could opt for one? The design research is still being done, I know, but I just wanted to

suggest that we might want to carve out a little latitude in our group thinking ... though it threatens to resurrect the group "type""[34]

1.2.

"What I understand to-date is the following:

Each of these things are separate objects that nest together like a nesting doll -- sites being the biggest doll.

A person can belong to many groups and many sites. A group can belong to many sites.

A person has a profile page(s) that he can edit and others can view.

A group also has a profile page(s) that the group leader(s) (for lack of a better term) can edit and others can view. This is not a site.

A site is, well... a site."[35]

1.3.

A use case collection was initiated to explore "how students and faculty need to find and connect to other people", possibly relying on group profiles. The collection received three entries.[36]

2. What if groups were also primary constituents, like sites, being joinable, discoverable and having their own tools and content?

The suggestions in this group were exploring the implications of letting groups take the place of sites. It would be groups, not sites, that have their own tools and content, and users could search for groups rather than sites. Some suggested to replace sites with groups, others wanted to make space for a special kind of site where group identity was more important than site content.

2.1.

“I found myself in the curious position lately […] arguing that Sakai really doesn't create groups. It creates spaces (or sites), and then grants access to an ad hoc collection of people as a secondary matter. The definition of the group amounts to "Whoever can access this site right now," and there is little user experience of these groups persisting beyond the context of the site. By the same token, a relatively weak sense of group identity.

What if that got turned around, and the people came first? What if site setup were simply the last step of group setup? Perhaps that won't work out well, but it seems something worth pressing on.”[37]

2.2.

“*I think it's worth emphasizing the point - and it casts some of the issues into high relief - that there will likely be a *kind* of site which is really focused on a group identity like a club (and these will probably tend to be the public, joinable sorts) rather than a mere group which is more an administrative convenience or accident of circumstance (e.g. we are all freshmen of the same department required to take this course this term).

The collection of "group sites" thus envisaged would be in the minority. Put in other terms, "group sites" might have a social networking connotation rather than an administrative-technical one.”[38]

2.3.

“In other words, would it be possible to make *communities* first-class citizens, and to selectively add Sakai-supplied capabilities and Sakai-hosted content to the

online community? Since most paths in a collaborative web application will be associated with some sort of membership list, there would be no special need to distinguish "top-level groups" from "site memberships": a site membership *is* a top-level group." [39]

3. There is a conflation between groups and sites. What if we simply thought of groups as lists?

Sharing and group management were connected with the idea that groups appear as lists. Related explorations were suggesting that groups could be understood as labels inserted into a hierarchy: they can belong to higher level descriptive categories, and other groups can be categorized under them.

3.1.

Along the same lines, the notion of groups as informative lists was explored in the context of managing the availability of a person's profile information. Keli Amann created a mockup that outlined a choice between groups, connections and courses for the purpose of selecting the audience of the profile (Figure 5.9). The selection was subsequently refined with three types of groups: voluntary groups, organizational groups and classes. Each category contained a listing of the relevant groups (Figure 5.10).

Figure 5.9: Keli Amann's first round of design for groups as lists in access management[40]

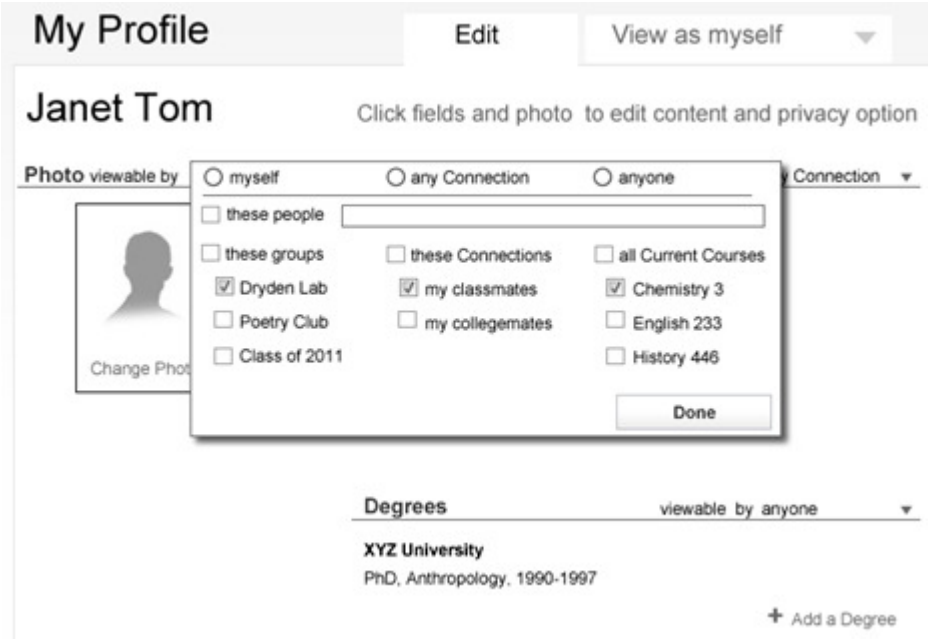
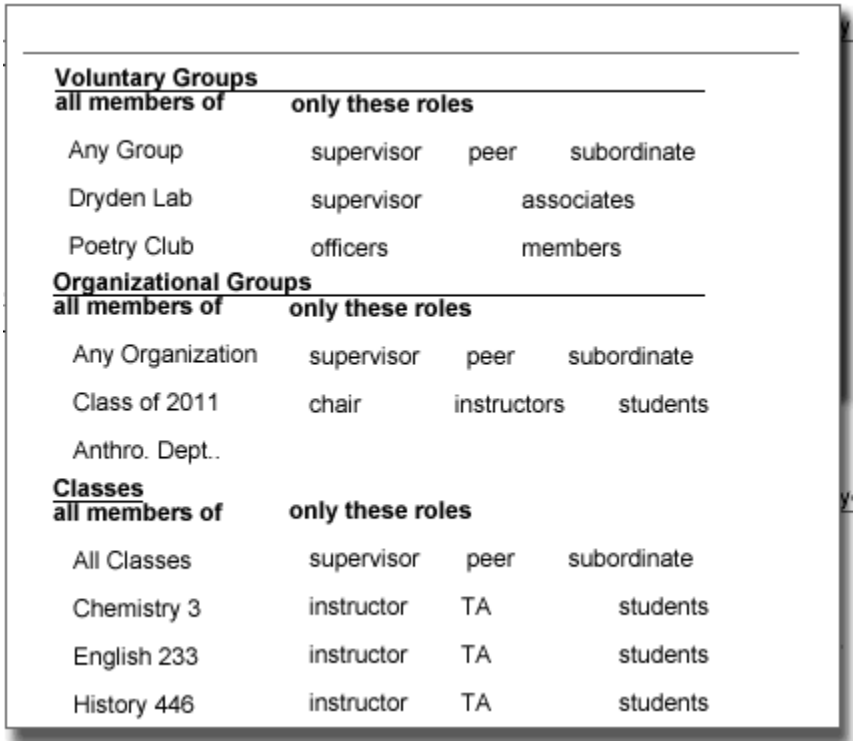
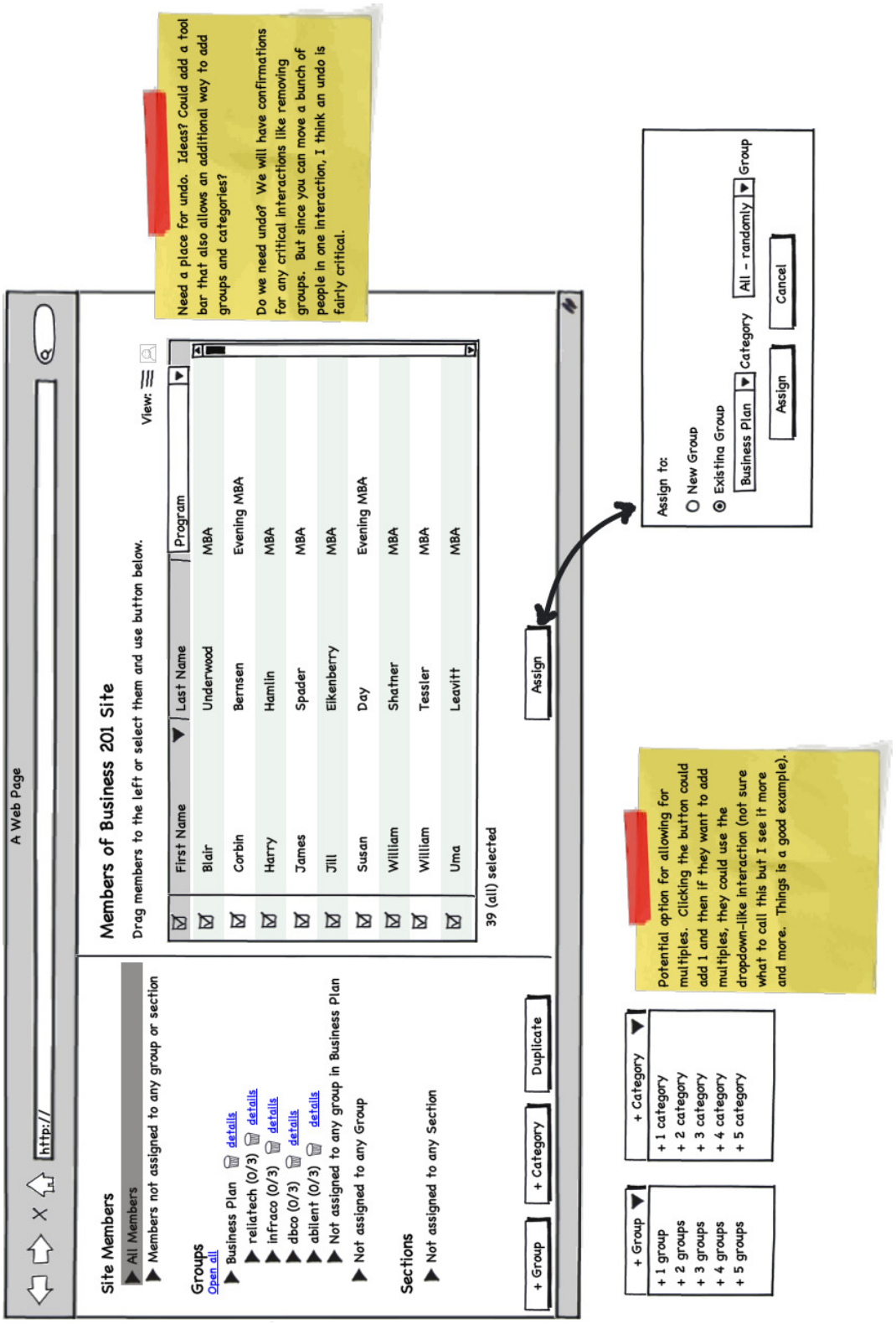


Figure 5.10: Keli Amann's second round of design for groups as lists in access management[41]



Voluntary Groups all members of	only these roles		
Any Group	supervisor	peer	subordinate
Dryden Lab	supervisor	associates	
Poetry Club	officers	members	
Organizational Groups **all members of**	**only these roles**		
Any Organization	supervisor	peer	subordinate
Class of 2011	chair	instructors	students
Anthro. Dept..			
Classes **all members of**	**only these roles**		
All Classes	supervisor	peer	subordinate
Chemistry 3	instructor	TA	students
English 233	instructor	TA	students
History 446	instructor	TA	students

Figure 5.11: A group management wireframe created in the design exploration in the Groups project[42]



3.2.

The Groups project came back to exploring group management workflows in a series of iterative design sessions. This effort was considering groups as hierarchical lists, which could contain groups or individuals in their listing (Figure 5.11).

3.3.

Some were deriving the idea of groups as list from a sense of discontent about overburdening groups with unnecessary attributes.

> "When thinking about ad-hoc groups, the main question I had was: are groups simply one's own contacts or are we talking about a place in Sakai where one can join and become a member?
>
> If it's the former, than the idea is fairly straight forward and serves as a convenient way to invite one's contacts into a site or a sites features/content, or to share a photo from one's personal file system, or an event in one's personal calendar, etc.
>
> If it's the latter, then I can see there being some confusion from a user's perspective (particularly those who haven't labored over this idea as much as we have) about the difference between a site and a group. Now we can do our best to frame the UI of the group space to be less functional so it's not confused with a site, but I worry that it's just a matter of time before we lose that discipline and start adding an announcement feature, or maybe a discussion feature, and so on -- before long, it is a site."[43]

3.4.

> "We can ask for membership lists from a corporate office, a psychology lab, a research project, a university class, a university student cohort, a collaborative

online space, or a video-sharing website. We can separately ask for the list of students in a class and the list of instructors in the class, or we can separately ask for the membership of each discussion section in a class. We can ask for the list of people who included an event in their calendar. That doesn't mean that "an event is a group," or "a website is a group" or "a website is a group of groups," or "a course is a group" or "a course is a group of groups." And it certainly doesn't mean that "a course is a website." It means humans are social creatures, many of our concepts include the notion of membership, and we sometimes want to use existing membership lists in a new context. […]

Based on past experience and some of the use cases and goals we've heard, I've started to think of "group" as not a thing in itself but instead as an *aspect* of lots of things. "Group management" takes place when I focus on that particular aspect. (The problem is similar to "resource" and "resource management." "Resource" is annoying jargon which hints that I occasionally want to put handles on something and move it around without worrying too much about what it *does*. "Group" means that there's a list of people that I at least occasionally want to think of as "members" of something.)"[44]

The thought experiments and design explorations emerged as local attempts in the web of discourse to construct a new conceptual model for groups. At the heart of the new model there was often a comparison to a well-established model:

1. A group is like a person, insofar as a person has a profile that can be used for finding them or reaching out to them.

2. A group is like a site, insofar as a site has tools and content associated with them.

3. A group is like a list, insofar as a list is used to display, organize and pick items.

Some suggestions also made a comparison with a model that was known to be undesirable: a group cannot be like a site insofar as a site has tools and content because that has already been discarded as an inflexible solution.

The contributors were not proposing a new model, only a possible direction for one. Thought experiments in particular were formulated as an incomplete possibility, inviting others to participate in exploring the implications of a line of thought. Their perspective was focused on the suggested singular approach to groups, and they made no attempt to consider parallel accounts. Within the overall design of 3akai, the three views were formulated as contradictory and incommensurable, but no attempt was made to reconcile them. The complexity of the problem was clearly perceived, but it was not explored in a systematic manner across the entire range of possibilities. The different speculations existed on their own, within their conceptual space that they carved out for themselves, and they were never played out against each other. They were constructing a possibility, but at the same time, they tended to view that possibility as exclusive and definitive of the underlying concept of group.

There was a fourth group of suggestions of a different nature, with a focus on the variability of groups. They were trying to build a conceptual model for the dimensions along which groups may be different.

4. What if we thought about groups in terms of different group types?

4.1.

> "I think the distinction between different kinds of groups is important. […] Based on the type of group, Sakai could recommend they have a site (or not) with certain communication avenues and tools. […] If we try to understand patterns of different types of groups I believe we could make UX's that match those patterns. This is early thinking. I'm going to start listing some of the different kinds of groups we know about and what those patterns would be."[45]
>
> 4.2.
>
> "In trying to imagine what a group management dashboard widget might look like, I think that: the basic group is just a list of people. I think "send a message" to these people (or "send an invite" to someone not of these people) might be all the functionality I would require in this group management context. Some of these groups (and not all) might have opted for "group sites," and maybe there's a little icon or link next to these kinds of groups that I can click on to take me there. Some of the groups I may have management rights to, and I'm given some additional options about those. Some groups might not have managers at all in Sakai, since they are externally maintained (e.g. from LDAP or registrar data). They're just there for convenience when I do other things, like share content or send out a message."[46]

5.1.4. Going beyond initial group conceptualizations

The possibility of types of groups suggested in the fourth conceptualization could have meant a possible way out of the contradictions between the various group concepts, allowing for profile-owning, site-owning and list-based groups, but it still could not resolve the contradictions inherent in their combination. Doing this required that

participants revisit the content model of 3akai and investigate how the different group concepts played out against it. This approach allowed the emergence of novel user interfaces based on a new model.

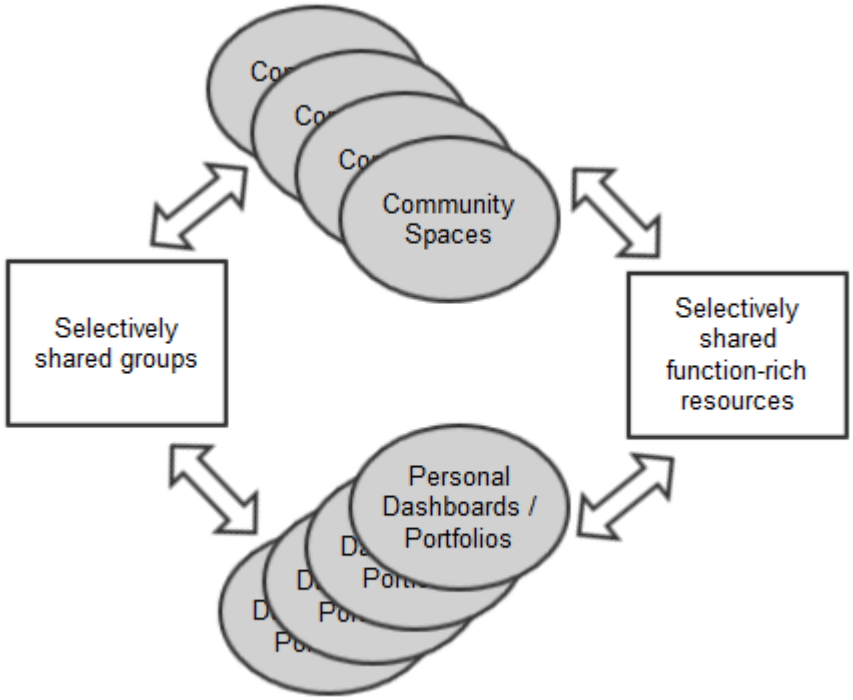
The moment of shift did not occur once, it happened on at least two, but possibly three distinct occasions, which did not influence each other. Thus, the new UX approach was invented multiple times before becoming the community approach. In one case, the user interface appears to have followed the grasping of the new model, in another, a reverse process can be glimpsed. I will describe each episode separately.

1st episode

In August 2009, around the time when the Groups project was initiated, one of the project's active contributors created a confluence page outlining the idea of a community space. Community space was illustrated by a diagram, reproduced as Figure 5.12 below, which placed groups in a visual parallel with content. The caption read "**Breaking the silos**" as a reference to Sakai 2's tool silos of content, and the figure displayed "groups" and "function rich resources" (i.e. functionality and content) symmetrically on the left and right side of "Community spaces", showing them equally severed from it. The image contained a second, horizontal symmetry between "Community spaces" and "Personal dashboards/portfolios", suggesting that community spaces are group-versions of personal dashboards and portfolios, and they are both associated with groups and content in a similar way.

Figure 5.12: Ray Davis' suggested new conceptualization for groups, sites and content[47]

In the author's language, community space stands for site, and resources stand for content.



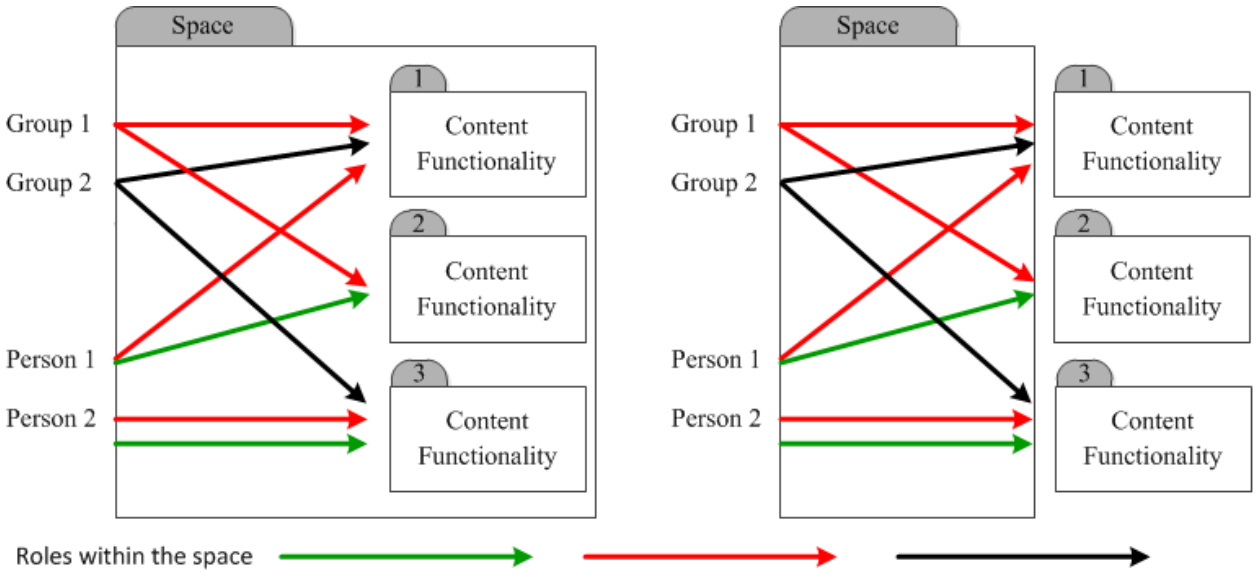
The related segment of the text read:

"So here's a thought experiment: At the top of the Sakai site are Individuals and Communities. Individuals have a "My Dashboard"; Communities have a "Community Space". In "My Dashboard" I can see what communities I belong to with links to their spaces. A known person doesn't have to "create a site" to have a space. A known community or organization doesn't, either."[48]

In many respects the model was vague and difficult to decode. There was no interpretation given for the meaning of the two way arrows used, for the plurality of grey

circles in case of spaces, and no similar plurality for groups and resources. At the same time it was suggestive of a conceptual model with a dashboard space, which somehow connects content and groups, but does not contain either of them. One may even argue that the model contained a contradiction. It was suggesting that groups and content exist independently of community spaces, but the only reason for this independence suggested in the picture was for them to be used in different kinds of spaces, like personal dashboards or portfolios. In other words, the diagram did not indicate a place in the user interface where groups and content could be accessed on their own, independently of spaces. Thus, the model was clearly unfinished, but it was suggestive of an overhaul of the content-model guiding 3akai. Figure 5.13 shows my diagram for this new conceptualization against Korcuska's model.

Figure 5.13: A comparison of my diagrams for Korcuska's version of the content-model and the group-model



The model was formulated in full in a comment made to the page by Clay Fenlason. Clay described the possibility of having activities without a site:

> ""At what point does a group want a site?" Or, put another way, what is a site good for? I think our answers to this one have often been wrong.
>
> In Sakai 2 you need to have a site to have a group. Pages, membership, and activities are all lumped together into a small container. […] In our talk of Sakai 3 (and even the UXI wireframes) I think we often imagine that one needs a site in order to have **activities**. I think that's also wrong. […] the activities of most groups, including the most common course activity, requires nothing like a 'site.' We should therefore be wary about introducing the site concept and its attendant management functions." [49]

He then went on to draw some possible implications of the new model. He first distinguished between two types of user interfaces, a group profile and a structured content site. The latter was described in terms of the original content model, which required content and functionality to be associated with a site:

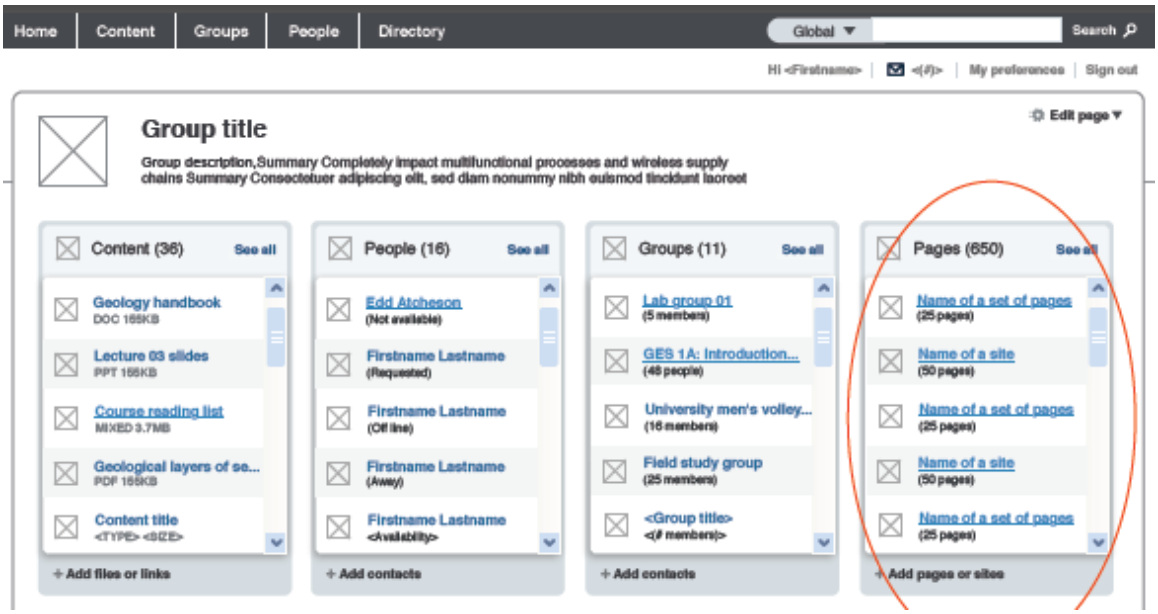
> "[T]he chief value of a 'site' is found in structured content. The fact that activities can be arranged around that content is incidental to the site concept.
>
> [T]he 'workspace' is an inadequate replacement metaphor for 'sites,' insofar as it conflates activity with content, and assumes that you need to have a 'space' before you start doing things."

To give a sense of the new group-based user interface, the author suggested that the group space was an emergent output of activity:

> "What if we **did things** first, and the site simply emerged from the output of whatever it was we started doing in our groups or individually? What if activity preceded the site, and there was no 'site creation' as such? Why do I have to carve out a space before I start acting?
>
> Let me put my earliest question yet a final way: "What are all the things a group can do before it gets a site?" And two followups: 1) "Can we design a good UX around these activities without having to think about workspaces?" 2) "Can these workspaces emerge as a living record of doing these activities?""[50]

Incidentally, he also dissociated roles from sites, and tacked them onto groups.

Figure 5.14: Implementation ready screen design for the new group dashboard[51]



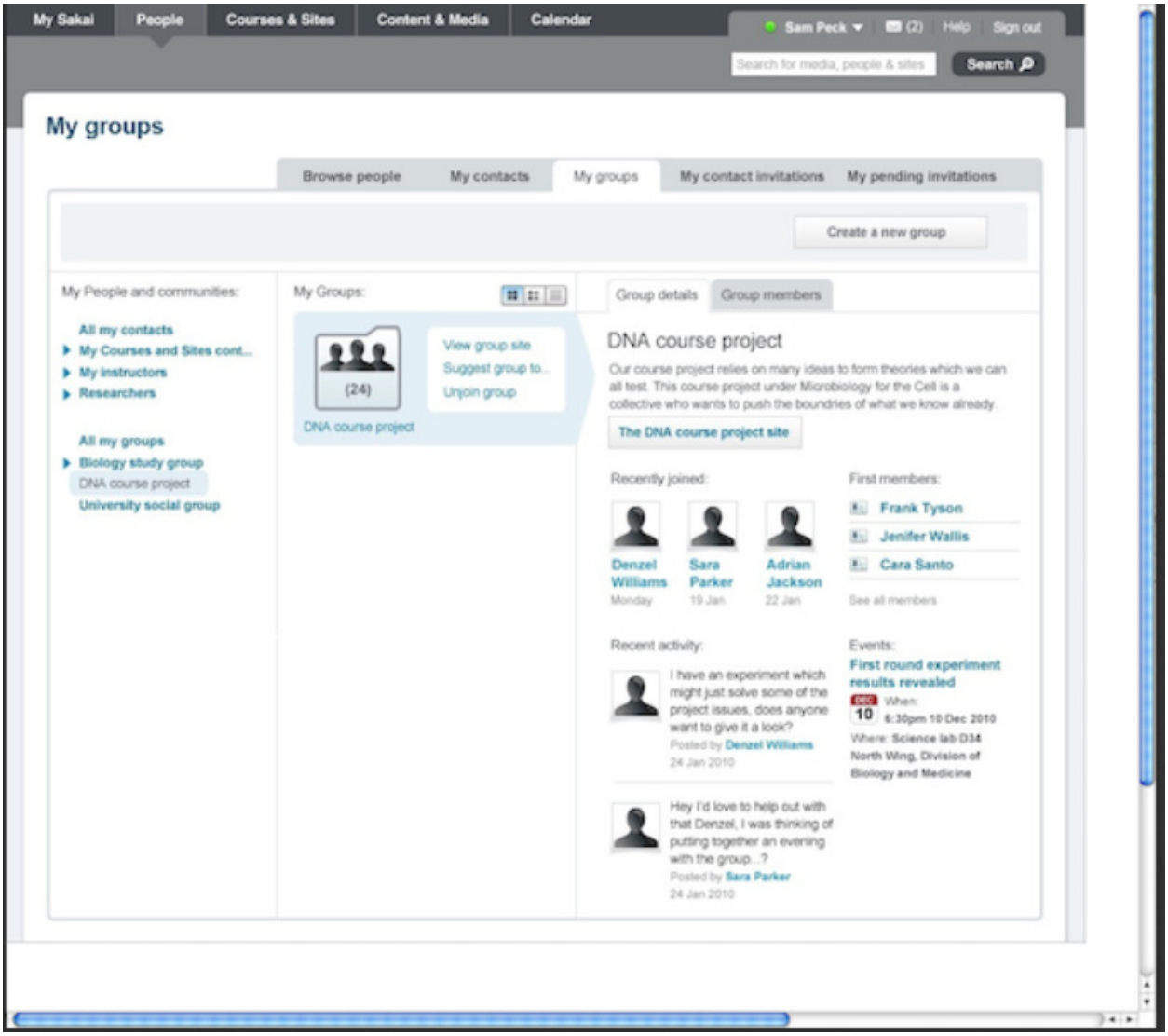
The result was a double model, with a content site made from content widgets, and a group space made from activity widgets. Figure 5.14 shows a user interface mockup of the group-dashboard with widgets showing activity related to the group in

connection with the four categories of content, people, groups and pages. Instead of structured content, the group site displays items associated with recent activity.

2nd episode

In the second episode, John Norman shared a recent insight on a community email-list in early 2010:

> "There has been an exchange on list about groups. Although I am going to refer to a concept that emerged only on Friday (i.e. 2 days ago) from Sam's [the consultant's] new work on Groups and I am not sure it will not lead us around in a circle, I thought it might be worth sharing. In my mind it starts to make clearer what it might mean to the user that we conceptually separate group membership from site membership.
>
> First take a look at this screen:



As soon as I saw this I thought it 'felt right'. […] I don't know how this should be hooked together technically, but I was struck that we could extend the concept with a simple comments wall (and maybe more) without necessarily creating the idea of a 'site' in the mind of the user.

The idea that a group has a 'site' as a set of web pages also seemed much more natural. When you get to those pages, who can see and do what may depend on their being a member of the group and their role within the group, but the site is a separable concept."

In a subsequent response he provides more clarification:

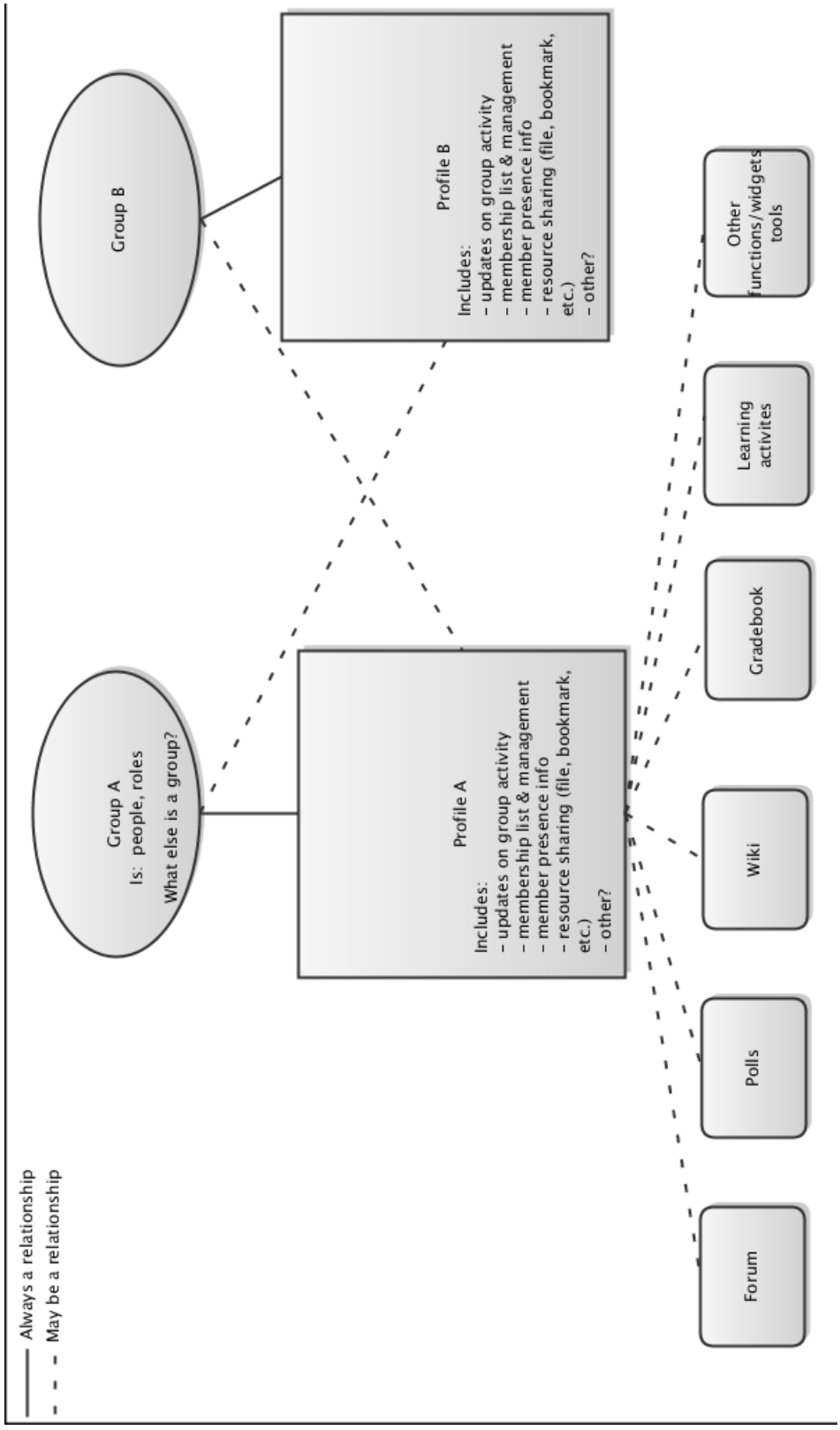
> "Until I saw this I had interpreted a 'group without a site' to imply simply a list of names. This screen is richer, pulling together a variety of information for the group members.
>
> […] This model has just popped into my head, I don't know if it helps:
>
> Course Site (today) = People, Content, Tools
>
> changes to
>
> Course: People and interactions (required) - browsable/searchable in People Tab
>
> Course: Content and interactions (optional) - browsable/searchable in Content & Media Tab
>
> If I want to find people to form a study group, I go to People Tab and I can see people who are in my courses/sections as well as anyone else in the University (e.g. all Physics first years). If I want to look for Thermodynamics teaching materials, I go to the Content and Media tab and I can see which materials are in my courses and which are from other courses (or even Research Groups).
>
> The course site (pages) becom[e] optional."[52]

In this second episode, the rearrangement took place in the realm of the user interface: John Norman saw the conceptual model of a new type of group space in the user interface which pulled together dynamic information about activity associated with members of a group. This appeared as a UI translation of Clay Fenlason's notion of activity history. The new user interface was formulated as an alternative to the site built around structured content. Norman also came to the conclusion that roles resided with groups, and groups could have a content-site besides the group dashboard, if needed.

3rd episode

The third episode has little process documentation, and thus we may only speculate that it emerged independently of the previous one. Alternatively, it could have taken place as the purposeful refining of the group dashboard user interface. The episode was centered on a series of diagrams which resulted from discussions between participants of the Groups project in early 2010, about a month after the email presented in the 2nd episode. The diagrams describe the user interface in terms of relationships stated visually and textually, and they build up a complex world of UX possibilities starting from the basic case of a group having a profile. The group "is people and roles" (in the third diagram: "the community is people, roles and subcommunities"), and it is "always related" to a profile which includes: "updates on group activity, membership list and management, members presence info and resource sharing". The group profile can have a relationship to different functionalities. The second and third diagrams (not reproduced here) shows optional relationships for both groups with a workspace/site/pages/page collections, which can display widgets, and indicates that the workspace may have a relationship to different functionalities.

Figure 5.15: Conceptual diagram of relationships in the user interface[53]



In the S/OAE project, the new model received the following UX design for implementation:

Figure 5.16: UX-Implementation ready design for the new group dashboard, containing the preview of the site for the same department (upper left widget).[54]

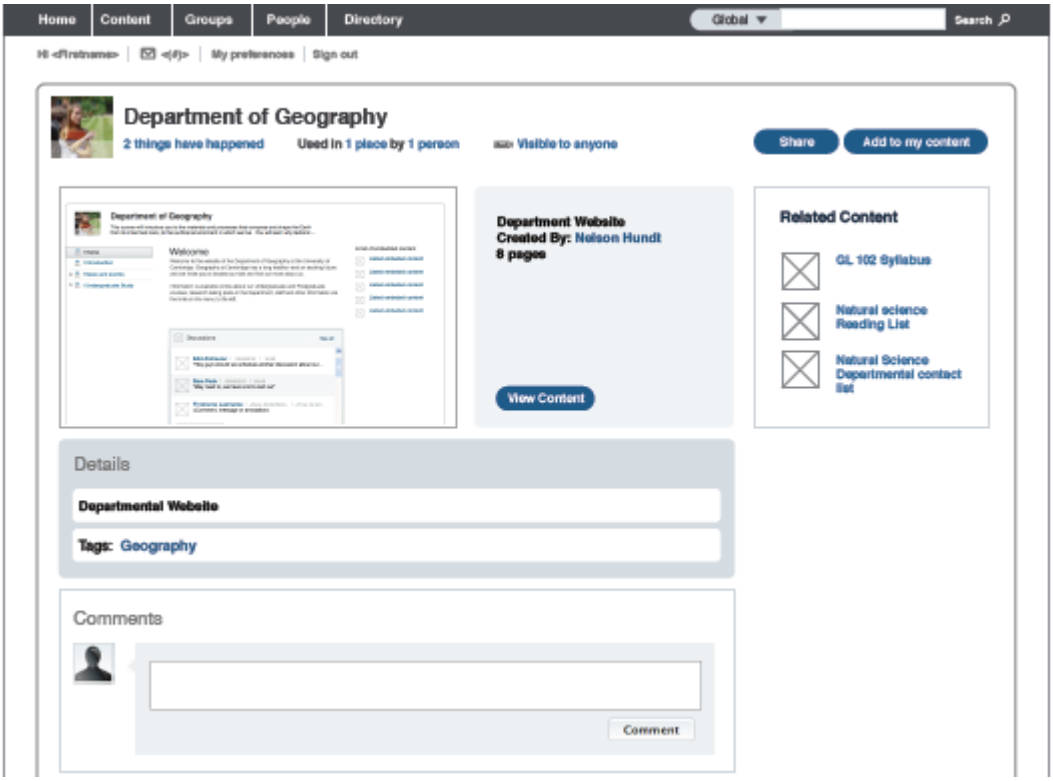
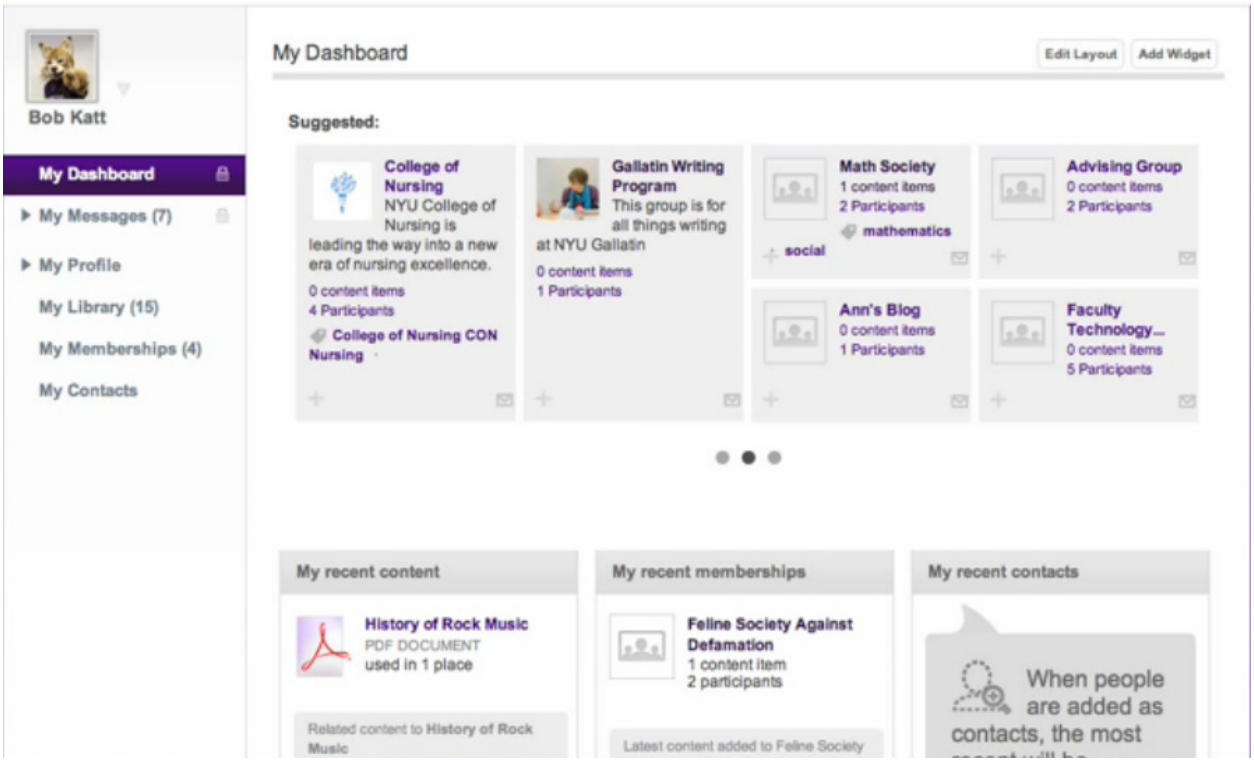


Figure 5.17: Screenshot from the video of the 2011 (2nd) NYU-pilot of S/OAE, showing the standalone availability of basic functionality from the personal dashboard.[55]



In response to the new design, the kernel development team dropped the site module that had been developed for 3akai, and created the separate groups and pages modules in S/OAE.

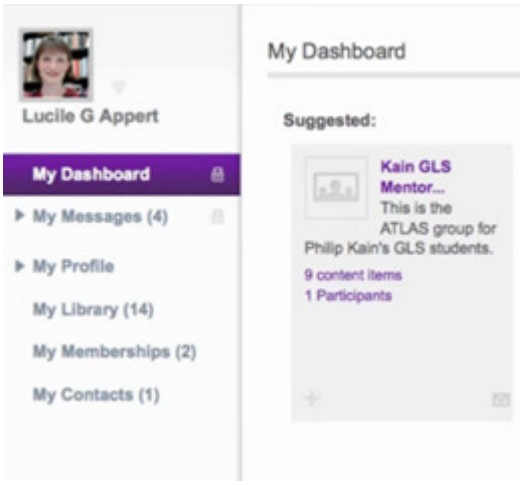
The double model solution created a new conceptual context, where other open-ended questions could be settled. One of these was the clarification around lists. After the new user interface had become accepted for implementation, one of the participants suggested to think of groups as two-way interactions between people, and lists as one-way interactions. List was defined as “the kind of thing you might use as a target for other actions: identifying recipients for a message, or granting a set of people a common

permission. A list is something an individual might create for themselves as shorthand for addressing a larger set of people."[56] This approach reconciled earlier group management plugin approaches with the conceptual groupings implied in lists.

The question of activity without a space also became settled. The group-dashboard contained links to areas where interaction with different types of content could be pursued. One of these areas was content management and sharing, another was messaging. The central idea of the group dashboard model was that these areas were not located within the group space, so content and messaging both received their own space, which could be reached from a link in the system. It may also be argued that the group dashboard opened the way for a high-level conceptualization of Sakai 3 made up of many different kinds of "spaces". The NYU pilot implementation had five different types of space-like areas, which could be accessed from the main menu (see Figure 5.17 and V.18). Note that no course is listed in this example):

- The entry page or landing page (My Dashboard)
- Course site with structured content (Course)
- Group site (available from My Memberships)
- Personal profiles (My Profile, further profiles available from My Contacts)
- Tool spaces (My Library for files, My Messages for messaging)

Figure 5.18: The menu of the NYU pilot implementation[57]



5.2. Discussion

In the above I have shown that the design of Sakai 3's group dashboard was the output of a lengthy process of conceptual modeling with several contributors. The process was distributed over time, across several participants and involved an array of artifacts. I will now turn to the discussion of the process from the perspective of the possibilizing and generative aspects in design, and how the temporal and social distribution of the process played into this. I will return to the third aspect of distribution in terms of artifacts in the next chapter, where I will describe the role of prototyping and the role of prototype artifacts as framing devices.

The process of conceptual modeling I have described unfolded over time, but it was not designed, planned or managed. Rather than following a previously established goal, it was unfolding counter to specific goals that had been established in the various projects. Groups did not figure significantly in the early threads resulting in Sakai 3, and the concept was not a part of the conceptual model of Sakai 3 at the time. Groups made

an appearance in the course of engaging with the details of the user interface. After a brief episode of cognitive engagement with the concept, it was dropped from the official agenda on account of complexity.

The various thought experiments and design explorations did not come about in a planned and orderly process. They were motivated by the open-ended character of the design space, and its way of framing openness for the purpose of design, and they could emerge in the midst of ongoing discussions. The Sakai 3 vision brought together fragmentary facets of a future system under the coherent, but high-level framing of the web 2.0 approach. On the one hand, the resulting design space was constructed in a way that invited interpretative engagement: the formulations of the Sakai 3 vision and the various prototype systems were suggestive of a conceptual space without being specific about the details. On the other hand, the design space was suggestive of unity and coherence. As I have argued previously, this unity came from the application of the discourse of the web 2.0 computerization movement, but it was seconded by the projection of a future system.

Thought experiments have been well known and well documented in the sciences (Shepard 1988, Nersessian 1992b, 2008a). Related to the role of "what if" type of thought experiments in scientific reasoning, Trickett and Trafton (2007) have argued that these are used by scientists in situations of informational uncertainty, when the informational resources that would be needed to think through a problem are limited. Simulating imaginary situations on the basis of what is already known from earlier experiences helps resolve the uncertainty of the situation. Nersessian (2012) has further shown that experiments in engineering labs represent a cognitively distributed construction process,

whereby researchers construct conceptual simulation tools to experientially study aspects of phenomena that are not directly available to experimentation. My analysis of conceptual modelling in the design of Sakai 3 also suggests that uncertainty fuels thought experiments. In the design context, this uncertainty may be encountered as fragmentary, tentative formulations of a future, projected artifact.

It may further be noted that conceptual modeling has long held a central role in software engineering (Kent, 1978) , and the construction of possible scenarios (generally referred to as use cases) has also figured centrally in requirements engineering methods, notably in conjunction with the use of the unified modelling language (UML) and object-oriented modelling (OOM) (Van Lamsweerde, 2009). At the same time, the theory of requirements engineering has not come to embrace the perspective of uncertainty and partial knowledge that has come to the fore in the analysis of scientific thought experiments. The main interest has been instead on formalizing related conceptual processes through formal modelling methods, to reduce the uncertainty in the process of formulating requirements. My analysis of conceptual modelling for Sakai 3 suggests, alongside parallel thought processes in the sciences, that requirements engineering would benefit from engaging with uncertainty and fragmentary knowledge that arise from the projected character of software systems in the making.

The sociology of expectations has also pointed out certainty and uncertainty as central aspects of the landscape of innovation; in particular, it has been suggested that there is variation in certainty across the actors, and over time. While uncertainty is normal in the immediate field of work, certainty among the supporters immediately outside of this field is important for keeping the project space together, and a spike of

uncertainty may be cited as the reason for the collapse of innovative projects (Van Lente, 1993). My analysis suggests that the projects unfold from the interplay of certainty and uncertainty, or the interplay of what is known and what is possible.

The need to make sense of the design space was expressed by participants at several times – note Nathan Pearson's admission of a lack of understanding, or the way the thought experiments were framed:

> "I'm not certain how best to account for this, but I'm starting to wonder …"
>
> "What I understand to-date is the following …" (Implying that there are other things that the person does not understand.)
>
> "When thinking about ad-hoc groups, the main question I had was"

Thought experiments were also tentative, expressing uncertainty in terms of the viability of the suggested approaches.

The interplay of what is known and what is uncertain played out in connection with concepts and their connections. Participants were driven by questions related to concepts, which could be typically formulated as the following: "What are (groups/course sites/roles) for the purpose of the Sakai 3 user experience?" Concepts in this context were for the purpose of user experience, which implied an answer in terms of activities with a user interface. Thus, a group could be accounted for as a list for the purpose of group management interfaces, or group could be people with a strong identity rooted in activities in the physical space, who would only use the platform for communication, or it could be people who rally around a common interest, and would use the platform for showcasing and nurturing that identity. The activities implied connections with other elements in the Sakai 3 user experience, such as pieces of content, structured content,

tools for engaging with content, widgets, panels of group and site management, search, and so on. User experience, as I will argue in the following chapter, projected its own coherence around these conceptual elements, and prompted an examination of further connections in terms of this coherence. The group concept could work well with some of elements of the interface, but broke down with the others. If groups are intrinsically a space with tools, how could they also be lists? If groups are no more than lists, that works well for immediate communication, but how can they be found?

However, the elements of the user experience were both known and uncertain to some extent. This was a consequence of the nature of the prototype implementation. There was an intuitive sense and a first implementation of widgets, but their full potential was only sensed. Similarly, there was an intuitive sense of search, file sharing, structured content, contacts, and many other things, which were partially implemented in the prototype, but not finalized. The conceptual models were working in this terrain, keeping things known and established “for now”, for the purpose of the conceptualization, while changing others. Their aspiration was to recreate the coherence projected by the user experience of a working software artifact at the level of the conceptual structures which were called to define that interface.

The uncertainty was further confounded by the socially distributed nature of the process, or what Scacchi has described as the web of discourse in open source communities. In the early phases of the design, when the conceptual modeling was taking place, there was no routine procedure for establishing shared knowledge, and participants came to the discussions with the assumption that others might know more than they did, and more importantly, certain things might have been agreed upon in some sort of a

community consensus they were not aware of. In this context thought experiments were the uncertain and tentative solutions seeking an element of certainty that may reside in the community.

Overall, the Sakai 3 design space worked as an epistemic setting where conceptual possibilities were played out. This process has been commonly described as exploration or search in the broader literature of design and innovation (Simon, 1996; Cross, 2000). While the term of exploration describes well the result that unfolded in the design space, it does not talk to what drives and frames the activity, and what accounts for the novelty of the outcome. In my analysis, I have suggested that the activity of conceptual modeling was driven by the possibilizing character of the design space which projected the coherence of a user experience over a terrain of conceptual uncertainties. This setup made space for a sense-making process by means of thought experimenting with various (or alternative) conceptual models, which prompted participants to bring elements of the user experience into play with each other, and eventually resulted in the construction of a novel conceptual model for the user experience.

CHAPTER 6. DESIGN FRAMED BY PROFESSIONAL METHODS

6.1. The framing of new technologies by the professions

Human-centered approaches have been grappling with their disciplinary contributions to software design. Advocates of ethnographic methods, who have been particularly vocal in this respect, have tended to emphasize the epistemic tradition from which they draw, and demanded place for this heritage within the processes of software design. They have primarily located the social sciences in the empirical nexus with social phenomena, and the role of theoretical conceptualizations has been explored as part of this broad framing, in connection with ways of knowing the social. At the same time, contribution to innovation has not been a central concern. In the coming chapters, I will be outlining a radical reframing of the challenge, which starts from innovation, and seeks to understand the possibilities of contribution to innovation in connection with the bounded indeterminacy of innovative design spaces, and the possibility of agency within this space. Thus, I propose to ask how the epistemic traditions of the social sciences may contribute to innovation in software, and how we may construct a strategy that can translate the social perspective into distributed cognitive-epistemic practices that are operational in the innovative context of a design space. The suggested strategy will be described over the coming chapters, which means that parts of the argument, and the conversation with the relevant literature, will be fragmentary and tentative before the final discussion. In the current Chapter I will investigate the notion of frames, and the framing of conceptual expansion in innovation by professional knowledge. Chapter 7 will look at a missing social conceptualization of the Sakai 3 platform in connection with its

perceived failure. Chapter 8 will explore how knowledge of the contexts of use enters into the process of conceptual expansion.

The influence of epistemic perspectives rooted within larger social groups has figured centrally in social accounts of design and innovation. The various approaches have been seeking to grasp the clusters of interpretations that enter into technological change, and the concept of frames has been used as the epistemic correlate of social groupings associated with innovation. Framing has emerged as a powerful conceptualization for the social constitution of the expansion of meaning in technological change, where frames may be understood to be providing an epistemic direction to innovation. Most notably, technological action frames have been shown to be at work in computerization movements (Kling & Iacono, 2001). Frames have also been linked to professional groups. Kling's early research (1980) showed how perspectives like automation and organizational efficiency were associated with the managerial professions. Orlikowski and Gash (1994) also used the concept of frames to theorize the professional perspectives that managers and technologists bring to information technologies within organizations. While SCOT has tended to emphasize the locally constructed character of interpretations, the social constructivist understanding of frames as "current theories, goals, problem-solving strategies," has also allowed for the accommodation of the epistemic baggage of the professions (Bijker, 1987).

Researchers within HCI and CSCW on the other hand have spelled out the foundations of their own human-centered and social contributions to the enterprise of making software in close connection with their argument for the relevance of the disciplinary traditions of the social sciences. The argument has been talking to

differences in professional framings, and specifically the importance of a human-centered perspective for software design, but professional contributions have not been considered specifically in connection with the epistemic content or direction of innovation. Lucy Suchman has probably come closest to this outlook (2002) with her recent notion of located accountabilities in technology production, which considers professional contributions through the lens of the plurality of knowledges informing design.

In this chapter, I will attempt to outline an account of how professional methods and tools may be understood to lend direction to conceptual expansion and to frame technological innovation in this manner. My case study describes the application of a user experience approach to Sakai 3 development. I will describe the epistemic contribution of prototypes as framing devices implicated in a broader epistemic strategy for shaping design. I will show that prototypes do not necessarily become directly implicated in conceptual modeling, but they can act as powerful framing devices by projecting conceptual coherence to the evolving software technology from a particular perspective. Related to the process, I will show how framing devices are able to stand in for the artifact, and I will describe two cases where the failure of professional approaches to influence the direction of innovation may be traced back to the lack of framing devices which would have been able to stand in for the evolving Sakai platform.

6.2. A threefold strategy for the construction of meaning

I have previously suggested that the new design space prompted participants to engage with the openness of Sakai 3. Their strategy of understanding could be seen to be relying on three pillars:

1. An emphasis on prototyping as a means of making space for exploration.

2. Relying on professional practices of user-centered design.
3. Learning from the contexts of use for feeding conceptual construction.

In this chapter, I will address the first two strategies. Prototyping was central for perpetuating openness and supporting conceptual exploration, while the design tools borrowed from the professional practices of user-centered methods were framing the process by projecting conceptual coherence in terms of the user interface. I will emphasize here that user-centered methods were not directly responsible for conceptual exploration. On the contrary, they tended to be used in a convergent manner, with a purpose of creating closure. At the same time, the tentative artifacts created in the course of using structured methods became taken up in the open source web of discourse, and served as framing devices. It is in this quality that they reinforced the UI-based coherence of Sakai 3. User interface prototypes entered into the conceptual modeling I have described in the previous chapter, and they provided an implicit high-level framing of "what was being built". Even when participants were not directly engaged with the creation or review of UX prototypes, their modeling remained confined to the conceptual space delineated by the user experience paradigm.

I will expand this analysis later with the third element of the strategy, showing how participants were pulling in understandings about the external contexts of use, and how conceptual modeling was taking place within the web of discourse that emerged from these collection efforts. I will also argue that overall, the threefold strategy resulted in a learning-driven design space which can be accounted for as an evolving distributed cognitive system similar to the engineering research laboratories described by Nersessian and her colleagues (Nersessian et al., 2003).

6.2.1. Prototyping

Prototyping in design is customarily understood as the creation of approximations of an envisaged system in material form. Within Sakai, the activity of prototyping played a significant additional role: it contributed to bringing a novel system into existence.

One participant described the related challenge in his musings about the new community process as bootstrapping a system that was as yet non-existent:

> "What I've struggled with most is "bootstrapping" Sakai 3. That is, understanding the very first step in a process that assumes projects that are folded into a product – that there is some shape, texture, and flow that must not be disrupted.
>
> Essentially all of the language is built around "the release" or "the product", and our mental models draw from our experience to date. […] It is especially perplexing because it is absolutely clear that we have two familial products at different generational stages (one adult and one embryonic). There is presently no product into which the "Sakai 3 project" would be folded. This is a bootstrap project for a bootstrap product – a recursive paradox that unravels the whole process unless we find the base case.
>
> I suppose I might say: is it a chicken or is it an egg? When will we know and how?"[58]

In this account, the challenge appeared deeply related to open source practices, and specifically the history of the Sakai community: How can an open source community create a new software system out of its own past, when that past involves exactly another system? The problem was here formulated in terms of creating a project for a system that was not there yet, and calling it into existence by this very gesture.

Prototypes, and more generally, framing design activities as prototyping provided a way out of this "chicken and egg problem". Prototyping created continuity across projects by means of indexing ongoing activities to the past and the future. Sakai 3 was thus bootstrapped into existence by reframing Sakai 2-related artifacts as Sakai 3 prototypes, and simultaneously projecting a line of continuity from existing fragmentary manifestations of the system to a coherent future artifact.

2008 was the year when the Sakai 3 vision was firming up, culminating in the executive director's proposal of a future system to the community at large. In this period, Sakai 3 appeared as a potential artifact: it was there, and it was not. Participants in the two projects (User Experience Initiative and Content Authoring) initiated in this period thought of their contribution as directly entering the next release of Sakai 2, but possibly also paving the way for a new Sakai 3. Eventually, they did not make the planned contribution to Sakai 2, and their outputs were picked up by the 3akai project, the first in a line of projects that explicitly endorsed the Sakai 3 artifact.

The 3akai project marked the acknowledgement of Sakai 3 as a future artifact. Sakai 3 was painted as a completely new system, radically different from Sakai 2 and designed from the grounds up. 3akai was positioned as a prototype on the way to Sakai 3, which was relying on earlier prototypes created in the MySakai and UXI projects, and taking them further in the direction of Sakai 3, but having no aspirations to being Sakai 3. As one participant pointed out, framing development as prototyping created a threefold structure, splitting Sakai into "3 projects"[59]:

- ongoing work on existing software,
- project goals, formulated in terms of a prototype release,

- the definitive future Sakai 3.

While others contested that there were only two projects (a current project and one in the future around Sakai 3), the threefold partitioning was not entirely without foundations, as three different framings of ongoing activity could be seen at play. In this threefold structure, ongoing development was working with whatever had been produced up until that point by previous development efforts, and it was aiming toward a project-related prototype release, which was not yet identical with Sakai 3. This is how Sakai 3 as a future system was bootstrapped into existence on the back of evolving prototypes.

The threefold structure was in place for 3akai and the Simple Learning Environment, until the managed project was initiated. As it happened, neither of the two projects actually produced the envisaged "production-ready release" which could have been deployed in a real higher education context as a pilot with users. The first pilot release came out in the early stages of the managed project (in October 2010). This first release also marked the end of the threefold structure, which came to be replaced by carefully planned releases in short cycles (called "sprints"), focusing on the implementation of specific features.

Framing development with the threefold structure as prototyping made way for experimentation and growth within continuity. The 3akai project was in the most delicate position in this respect, because it was inhabiting a complex arena of existing and projected artifacts. At the time of the formation of the 3akai project, it was clear that it would be an experiment, and the release would be later replaced by a new solution relying on more adequate foundations. In many ways, this followed from the constraints of the situation. A new kernel (K2) for use in the future Sakai 3 was underway, but far

from being finished. Thus, it was agreed that 3akai will be built on top of Sakai 2's kernel, K1, which had also been used in MySakai and the UXI implementation. Thus, 3akai was conceived as a transitionary artifact supporting the exploration of a web 2.0 user experience for Sakai. It was considered useful because it should allow a deployment and user test in the coming summer, but it would be tentative and incomplete in nature, since it was expected to be followed by a K2-based system. As a contributor put it:

> "A milestone release for Sakai 3 has been set for the June/July time frame. This milestone release will be a prelude, and to some extent, a test case, for some of the ideas envisioned for Sakai 3 – but will not be Sakai 3! Sakai 3 will materialize with your help over the next full year (or more) by undergoing a proper, robust, and rewarding design process!"[60]

At times, participants had a hard time making sense of the various planes on which Sakai 3 was conceived, as repeated discussions on the subtleties of identity and difference suggested. What is Sakai 3? Is the system under construction already Sakai 3? How are Sakai 3 and Sakai 2 different, if they stand in a relationship of continuity in terms of an open source community? Indeed, why is not the system under construction Sakai 2? The community came back to asking such questions. In one of these discussions, participants were trying to find a name for a prototype server installation, and played with accounting for the transitionary nature of the 3akai artifact. The emails below were written in response to a suggestion to call the 3akai demo server a release candidate (RC) for Sakai 3.

Clay Fenlason:

“[…] let's not get ahead of ourselves. It's an implementation of the wireframes that came out of the UX Initiative, not yet even including visual design, and I, too, don't think we should confuse or presume by calling it an RC. In my mind a Sakai RC would have to show fuller course capabilities - the kind of thing someone might at least plausibly run as a pilot or be able to do some performance testing against. What we have here is mainly an interaction design preview, not yet even wired to the back end it plans to use. It's essentially just a working incarnation of Nathan's screens posted on Confluence.”

Peter A. Knoop:

“How about sakai3-preview, sakai3-dev, sakai3-in-progress, the-thing-that-might-possibly-maybe-someday-evolve-into-sakai3, or the-thing-we-do-not-want-to-call-sakai3-yet-but-there-are-no-other-alternatives-at-the-moment.sakaiproject.org?”

Stephen Marquard:

“r-and-d.sakaiproject.org

or

experimental.sakaiproject.org

Since we have a product development process, and the maybe-destined-to-be-Sakai-3 work is currently in the R&D stage.”

Anthony Whyte:

“dinges.sakaiproject.org

Flemish slang with various meanings depending on usage:

1. stuff 2. a thing for which you cannot recall it's name 3. a person whose name you cannot recall"[61]

Some of the suggested names, like "dinges", testified to the ungraspable quality of the work in progress system, others, like "sakai3-in-progress", or "the-thing-we-do-not-want-to-call-sakai3-yet-but-there-are-no-other-alternatives-at-the-moment" attempted to describe it exactly by its transitionary character. The "r-and-d" name indexed the system to the organization of the social context. Overall, all of these perspectives played into the understanding of the prototype system.

Meanwhile, in day-to-day engagement with development the different horizons were collapsed into a tentative, prototyping mode of work. This mode of work allowed for engaging with some of the details in the here and now, while it also outlined a conceptual grab bag where half-formed, unfinished ideas were stored and saved for the future. Conceptual modeling around groups grew from the fertile accumulation of this grab bag. Besides conceptualizations related to groups, the ideas that were set aside included search, aggregate widget displays, or the use of templates for providing specific structure over generic capabilities in support of novices and institutional customization. John Norman explained his perspective on these complex matters in connection with outlining his own version of the three perspectives discussed above (I have marked paragraphs quoted from a previous email with a right chevron):

">the current work

I think this is anything being done by people at Cambridge, GATech, UCDavis, UCBerkeley, Michigan, Indiana, etc. that is somehow expected to find its way into a Sakai 3 release. I believe it incorporates K2.

>the GT/Cambridge pilot

Mainly, the stuff above but there are some concepts (particularly in UX) that are being discussed and even worked on, that won't be ready for our use in the summer. Similarly, we have abandoned the compatibility layer for Cambridge deployment this summer. This term I understand to refer to the (diminishing) list of what we expect to be ready this summer.

>the definitive “Sakai 3.0”

This is some definition of the ultimate goal for a 'full' release that has not been specified. I believe it includes most of the current UX ideas as well as UX ideas that will be enabled by K2, but which we don't have time to implement in the summer deployment. For many people there is an expectation that 'the definitive version' will include migration options that differ from the limited plan we have for Cambridge this summer, but again they are unspecified and untested.” [62]

This account outlines a zone beyond 3akai, somewhere between 3akai and Sakai 3, where the “abandoned”, “unspecified and untested” ideas are relegated, including “concepts that are being discussed and even worked on”.

The conceptual modeling described in Chapter 3 grew out of this zone of ideas put on hold. At the time the 3akai project was formed, groups were considered central to the nature of project sites, and I have told the story of how it was set aside in favor of the dashboard design of structuring content, to reduce the complexity of the design challenge in the short term. Attempts at understanding groups for the purpose of Sakai 3 grew out of the web of discourse parallel to the prototyping activities. The conceptual modeling I have described was not directly connected to the actual development work in any way,

but it was framed with a horizon of contribution to some future version of the existing transitionary artifact.

In this manner, prototyping activity created the possibility of conceptual explorations in terms of its intricate interpretive tangle of current and future systems. Conceptual exploration was not relying on direct engagement or practical work with the actual evolving system; many of the contributors were no more than bystanders in that process. At the same time, as I will argue in the following section, the nature of the practical work with the concrete system entered into the framing of the conceptual efforts. Prototyping became implicated in the conceptual models based on a user-centered design approach. Conceptual models were seeking to be coherent in terms of user experiences, and specifically the various design approximations of the user interface.

6.2.2. UX-led design: tools from professional practice

The Sakai 3 vision made space for domain-driven design with an emphasis on user experience. A focus on improving the user experience for Sakai was clearly present in Korcuska's vision document, as well as in the web of discourse around it. There had been a perception in the community from the earliest days that Sakai 2 had been built in a technology-led development process. Some participants were explicitly seeking to make space for end-user requirements and the characteristics of the application domain in a traditionally software- and development-oriented open-source approach. The UX-driven approach emerged from these efforts, as a response to earlier practices in the Sakai community.

Table 6.1: Professional methods in Sakai 3 projects

The arrows indicate where outputs were taken up by subsequent projects.

	Central area of work	User-centered methods		Implementation prototypes
		Contexts of use	Artifacts of design	
3akai (UX Phase I)	What are content widgets?	Vignette use cases File-widget	Nathan Pearson's mockups	demo server Sakai 3 UX RC1 release
side project: Groups	What are groups in Sakai 3?	Group management use cases and scenarios	Group management mockups	people-picker interactive prototype
side project: Investigation	What are assignments for Sakai 3?	User research Personas and scenarios		
side project: Instructional visioning	Teaching & learning capabilities	Capabilities spreadsheet		
Simple Learning Environment (UX Phase II)	What are groups in Sakai 3?		Flow Interactive UX framework	demo server Sakai OAE Q1 Pre-release
Sakai OAE	How usable is Sakai 3?	User testing		pilot deployments
	What are the institutional requirements?	Functional minispecs with scenarios	UX-dev mockups	demo server Sakai OAE 1.0

Sakai OAE v.2	Performance	NYU pilot

In a practical sense, UX-led design was understood as an approach where design starts with the user-facing parts of the system: outlining how user activities are supported by a user interface. Participants translated UX-led design to their activities in terms of the following broad outline of the overall process:

- At first, the user interface should be designed with the help of mockups.
- Second, the mockups should be implemented on the front-end as an interactive prototype.
- Finally, the development of the back-end should derive its requirements from these implementations.

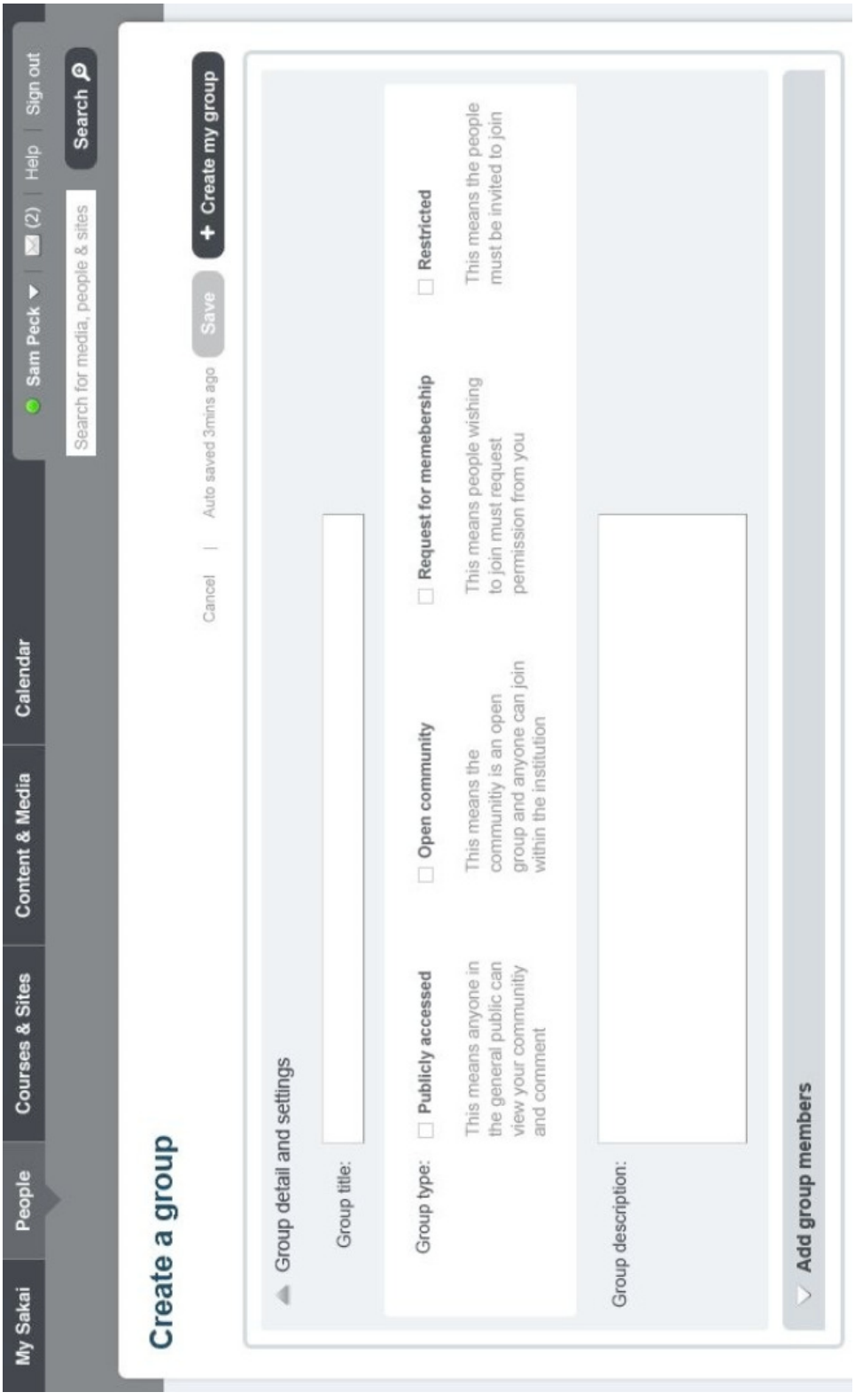
The approach was drawing on a range of professional software design methods, housed within the broad area of user-centered design. Table 6.1. contains an overview of Sakai 3-related projects with the various professional practices that were central in each. The arrows indicate where outputs were taken up by subsequent projects.

In the following, I will describe two areas of UX-methods that participants were relying on: prototyping and the creation of a UX framework. The various projects were also relying on UX-methods related to investigating the contexts of use, which will be discussed in Chapter 8.

Prototypes

Prototyping involves the creation of tentative versions of a future system in material form. Prototypes of the user interface included visual mockups, also referred to as screens or wireframes, as well as interactive walkthroughs created from these static mockups. Work in progress systems were also considered prototypes, as I have suggested before, and they also represented ways for participants to engage with the evolving user interface.

Figure 6.1: A screen mockup within an interactive click-through prototype, created by Flow Interactive as part of the UX framework



Prototypes of the user interface. Mockups for Sakai displayed parts of the user interface as a graphic. Typically, they represented entire screens, in which case they were referred to as wireframes or screens. In some cases they were pulled together to convey a sense of workflow. This could be a simple arrangement on a web page, with explanations or arrows, or it might rely on a basic html-framework which connected screens in interactive click-throughs. Figure 6.1 is an example of a screen from an interactive mockup.

Mockups were typically created with a software tool, which required familiarity with the tool, and in some cases further familiarity with graphic design. Because of this, many participants did not engage directly in the creation of mockups, but participated in the web of discourse that emerged around them. Most mockups for Sakai were produced by external consultants, who worked on various Sakai 3 projects over extended periods of time. Consultants' mockups were taken up in another common professional design practice, design review, where participants gave informal feedback. Design reviews inserted mockups into the open source context of the web of discourse. They involved considerable amount of self-organization: discussions took place in online meetings, written feedback was occasionally provided on Confluence or in email, and on one occasion, one participant's spontaneous invention of recording a screencast with audio was followed by many others, without however becoming an established practice beyond the particular occasion.

The UXI project hired a consultant, Nathan Pearson, who was tasked with a redesign of Sakai 2 user experience. He continued to work in 3akai. His main work method was the creation of non-interactive visual prototypes, which were discussed with

participants and adjusted in light of these discussions. These prototypes were to be turned into prototype-implementations, but the implementation remained fragmentary in the case of both Sakai 2 and 3akai. At the same time, they conveyed a focus around particular aspects of the user experience, which was mirrored in discussions beyond the design reviews.

3akai later hired Flow Interactive, and two consultants from this company intermittently continued to work on various projects until the release of S/OAE. As part of their work on the UX-framework discussed below, they produced a series of tentative mockups to probe the overall UX-logic of the new platform through community discussions. In the second episode of conceptual modeling described in Chapter 3, the double model of the user interface was formulated in connection with one of these mockups.

Besides consultants, some Sakai participants also created mockups. Typically, these were not intended to be final, implementation-ready designs, and often they were framed as explorations. Many community-produced mockups were exploring groups and group management, and they have been previously discussed in the analysis of group-related conceptual modeling.

It may be added that the role and status of mockups was not uniformly perceived by participants. Consultants tended to present mockups as final products, ready for implementation after a design review by participants. Participants, on the other hand, often took mockups as a starting point for conceptually engaging with design. With respect to these differences, the eventual contribution of mockups was settled according to the overall progress of the project. In early phases, consultant's mockups were readily

taken up and discussed in the web of discourse. They were also readily contested, serving as springboards for the formulation of alternative solutions. As Sakai 3 was moving forward, and experimentation gave way to expectations of implementation in the managed project, mockups were also received as more final. From the above it appears that mockups could equally be taken up in divergent and convergent conceptual approaches, and they were supportive of conceptual modeling as they became taken up in the web of discourse.

Work in progress systems. Live installations were used for two purposes: in ongoing development work, and as demos for the wider community. Initially, these functions were fulfilled by a single system, but as Sakai 3 grew, they were split up across several systems.

Developers needed running installations of the Sakai system as a context where new code could be made to work. Each developer worked with one or more local installations of the system for their personal use, where they could test their code. The code for these local installations could be pulled from open source code sharing platforms on a regular basis.

MySakai was originally made available as a demo at 3akai.sakaiproject.org, and the site gave its name to the 3akai project. When a chunk of code was considered ready, it would be uploaded to the 3akai server. Updates for 3akai were ad hoc, and often not announced, and the system contained a shifting set of broken elements that gave errors. The server was regularly visited by participants not involved in coding to experience the possibilities of Sakai 3 in a direct manner. It would also be used to demo the system to a wider audience, which mainly served to assert the potential of Sakai 3 as a future artifact.

At the beginning of 2010, three new servers were set up with different refresh policies serving different audiences:

- a demo server relying on stable code for "high stakes" demos: for showing the system to the Sakai community, and to the world at large;
- a prototype server with stable kernel code and the latest UX code, for project participants to interact with;
- a "bleeding edge" development server, with the latest kernel and UX code, for developers to test their code on a daily basis.[63]

Participants interacted with these various installations on a regular basis. At the same time, the installations were not cited in conceptual discussions about the new Sakai, which suggests that they served as a context to conceptual design, but they were not directly implicated in the activities.

The UX-framework

The UX implementation work was divided between two threads. One was the ad-hoc coding of specific features, the other was the creation of a framework addressing the patterning of the code. The framework was in part emerging from UX code, and in part from preparatory design work. The patterning of the code could take place at the moment of creation, as new structures were laid out to organize the work, or in subsequent systematic reviews, which attempted to bring order into what was already created. The review involved reorganization of existing chunks of code in hierarchical structures to support division of labor and parallel work across programmers. The most important part of the review work was directed at finding identical operations within the code base, and

merging them so that the operation would be located in a single instance of code. These code-related activities did not appear to be taken up in conceptual design work.

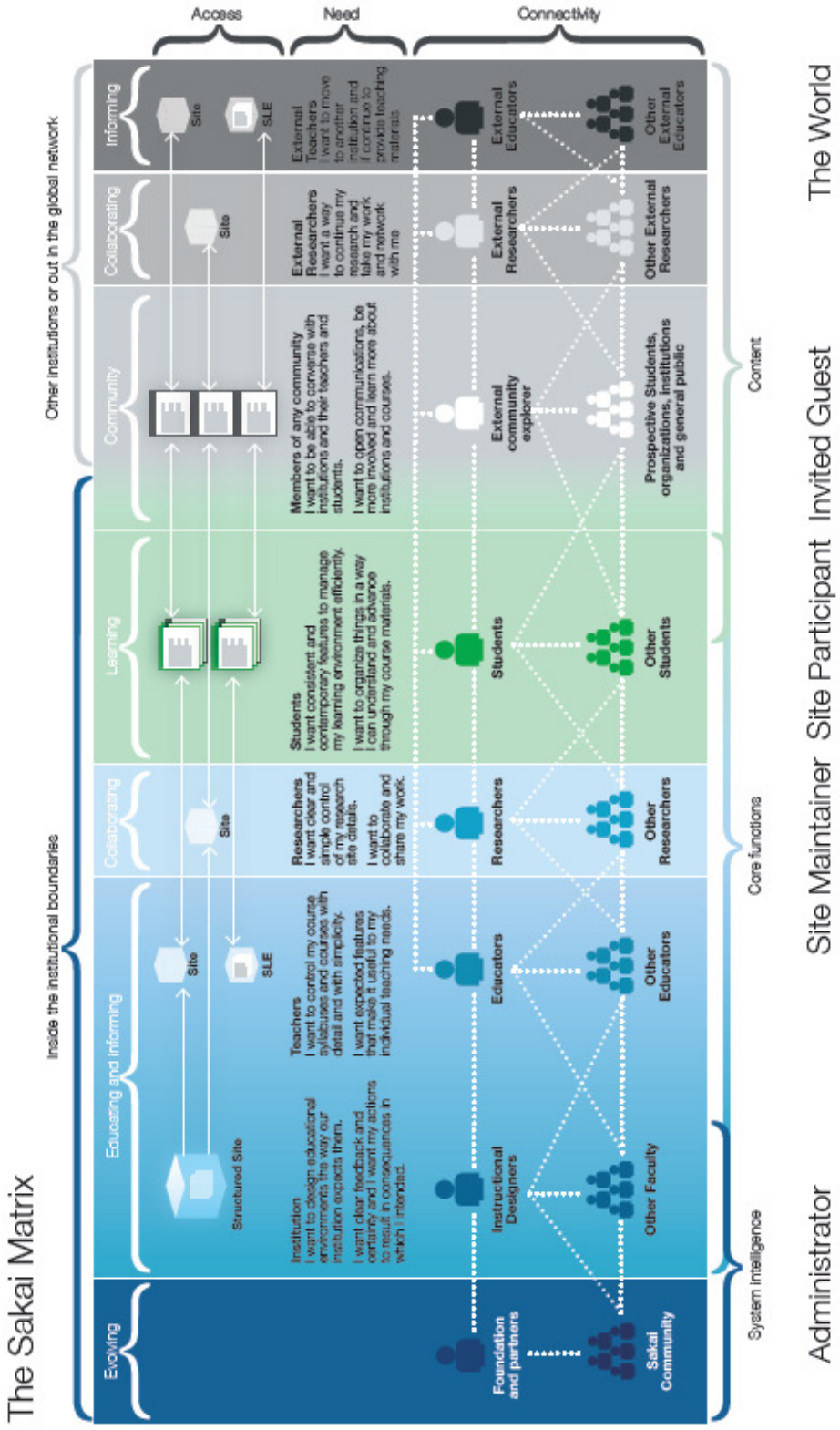
At the same time, the design of user experience was also investigating patterns to be followed by the code. In early 2009, Cambridge and GeorgiaTech engaged a design consultancy (Flow Interactive) to work on what was called the UX-framework, and the engagement was active until the first release of S/OAE. In discussions as well as in various graphic manifestations, the UX-framework appeared as the organization of the UX into meaningful workflows built from meaningful chunks of activities. At times it was expressed in conceptual overview diagrams, at times in interactive prototypes, and at times as plain discussion. There was also an understanding that the framework would eventually result in a design pattern library, mirrored by the patterning of the code.

The idea of the UX-framework was rooted in user-centered professional practice, specifically in the area of web design. The term itself does not refer to a specific method, but various ways of approaching conceptual order from the user's perspective of the browser interface.[64] As Garret describes his widely referenced framework, it "seeks to define the key considerations that go into the development of user experience on the Web today", and should be treated as open to new considerations emerging during the process of development.[65]

Within Sakai 3, there was no attempt to choose a specific framework, or to clarify the local approach. The UX-framework produced by the Flow consultancy was not taken up by Sakai participants at the level of concrete details. Meanwhile, participants came to describe their own efforts of conceptual organization as the creation of a framework. This

also involved the spontaneous efforts of conceptual modeling I have described previously.

Figure 6.2: A conceptual overview of user types in Sakai, created as part of the work on the UX-framework by Flow Interactive



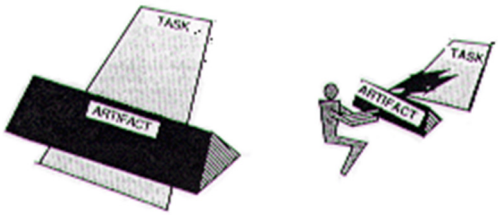
The conceptual models in the UX framework created by the Flow consultancy were different from spontaneous models in one significant respect: the UX-framework was seeking an orderly, synoptic coverage of the system from various perspectives, such as the universe of users in Figure 6.2, while spontaneous models remained fragmentary. In the example of Figure 6.2 above, the UX-framework appears to be operating with a synoptic grid composed of comprehensive categories, like types of users (e.g. educator, student, researcher) and types of activities (e.g. learning, collaborating, informing). Its categories make explicit what is assumed to be unstated and unquestioned knowledge in the Sakai community about the educational context. In this respect, it appears to aim at serving as a conceptual map of the terrain, with a purpose of explaining what is already established through the means of conceptual organization. In contrast, the conceptual model of the group dashboard interface emerged from efforts of sense-making, which were seeking to find novel connections. Overall, the UX-framework played a role that was analogous to prototypes, insofar as it was not directly involved in the expansive processes of conceptual modeling, but it contributed to its UX-based coherence.

6.2.3. User experience as a source of coherence in conceptual modeling

The conceptual modeling efforts described in Chapter 5 were seeking to make sense of groups for the purpose of outlining a user interface. While visual manifestations of the interface figured prominently in the process, the focus of design was broader than this, involving what Norman (1991) has called the system view of the artifact. Norman argued that the person involved in carrying out a task will experience the artifact as directly implicated in the task, but an account of how artifacts participate in the task requires a system view composed of the person, the task, and the artifact. This is the view designers

need to embrace. The distinction between personal view and system view is reproduced from Norman's work in Figure 6.3 below. Norman described the system view in connection with cognitive artifacts, and his formulation was made in terms of a system involving a goal-oriented task. Sakai 3 contributors outlined a system around the user interface, which included interface-related activities such as access to a site, creation, setup or editing of a site, sharing content or joining a group. I will call this the system of user interactions. Groups were implicated in this system, and the conceptual models around groups represented systemic views of the user interface, constructing a system of user interactions in abstract, generic terms.

Figure 6.3: The personal and system view of an artifact and the related task, reproduced from Norman (1991).

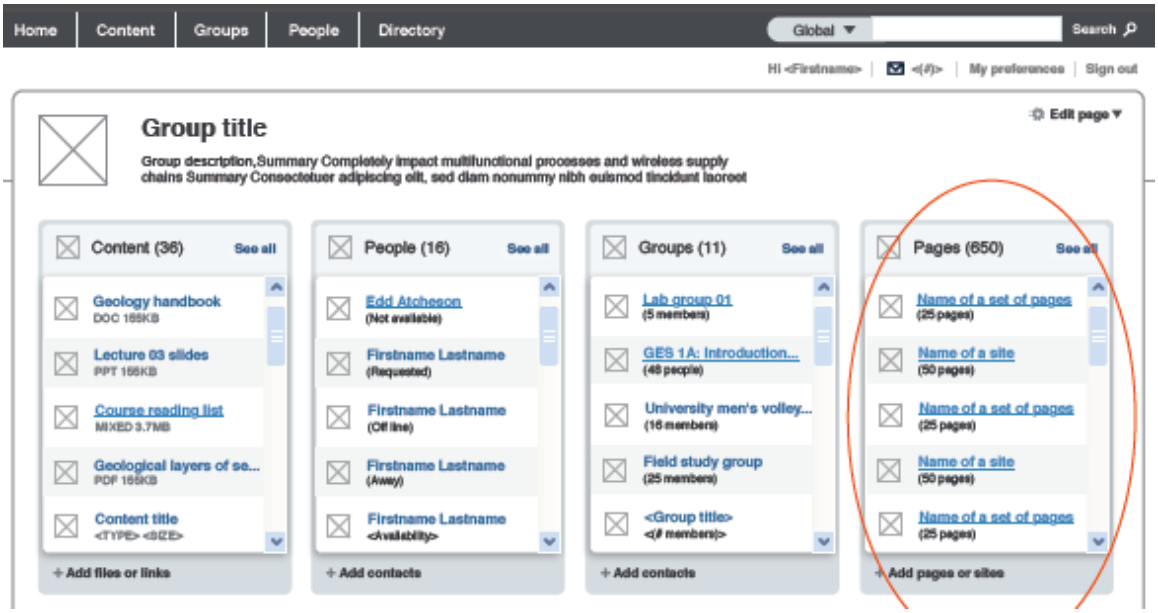


In user-centered design, prototypes are perceived to be central for the collaborative design of interactive software, as they allow for engaging with the artifact in a form which is tangible and meaningful at the same time (Bødker & Grønbæk, 1991; Beaudouin-Lafon & Mackay, 2007; Muller 2002). From a DCog perspective, it may be suggested that prototypes are taken up in the distributed cognitive process which underlies Norman's system of user interactions. It may further be argued that prototypes

allow for the social distribution of cognition across designers and users for the purpose of collaborating on various aspects of the design.

The analysis of the conceptual modeling process around groups suggests a more complex picture, with a chain of approximations. Most of the contributions in the conceptual sense-making efforts were using the secondary medium of language to depict an imaginary visual model, the computer interface. Contributors, it may be suggested, designed a generic interface in the absence of the actual manifestation of the interface, which was to be created subsequently. While we do not see into the heads of these people, the absence of forceful visual models in the context of conceptual explorations suggests that participants were working in terms of a multitude of private representations, which converged around generic conceptualizations of the system of user interactions. The eventual solution was a general conceptual model of groups implicated in a dashboard design. Prototypes of the user interface were indirectly implicated in this broader conceptual model. Participants were peppering their suggested accounts of groups with such UI metaphors as site, space, workspace, profile, dashboard or list. Mockups were at times used to convey such abstract relations as membership or subgroup in the visual language of the interface; for instance, in the second conceptual modeling episode described in Chapter 5, I showed that a mockup was helpful for conveying the new model. Following this episode, further mockups were created, like the one in Figure 6.4 below, to convey the generic idea that groups can have dashboards displaying digests of activity.

Figure 6.4: Implementation-ready screen design for the new group dashboard[66]



My account of prototyping activity also suggests that in the open source context, conceptual design grew out of practices associated with open source development, specifically the self-organizing distributed communication manifested in the web of discourse, and the emphasis on development continuity and prototyping. Professional tools and methods of user-centered design were not themselves driving the conceptual modeling efforts, but they had a significant role to play in the direction conceptual modeling was taking.

My claim is that professional UX approaches defined what was being built in terms of the conceptual coherence of the system of user interactions, and thus provided the direction of domain-driven innovation. Prototypes of the user interface played a central role in creating this coherence. I have previously suggested that contributors entered an open-ended design space created around Sakai 3. The Sakai 3 vision emphasized improving user experience as part of the overall strategy of the web 2.0

approach, but it was listed among other generic goals, such as cohesive learning experiences, academic collaboration, and the scalability and configurability of architectures. The design of a user interface became the focus of design work in connection with the MySakai and the User Experience Initiative projects, which emphasized the actual, concrete interface solutions. My Sakai suggested that dashboard interfaces represented a powerful approach in web 2.0 web design. This central notion was taken up in the User Experience Initiative, where it was given further support from user-centered design methods, with a focus on prototyping the visually manifest interface. The system was sketched as a series of interface snapshots connected into sequences of activity flows. On these foundations, the conceptual modeling efforts attempted to create a coherent experience around the interface. The novel understanding of a web 2.0 Sakai 3 arose along these lines, as a platform housing the practical logic of activities around a personal, course and group dashboard. It is in this sense that the novelty of Sakai 3 came out of a user-centered professional framing.

In this process, the various prototypes and the UX-framework played the role of framing devices, which contributed to the overall framing, but were not directly engaged with in the course of conceptual construction.

6.3. Unsuccessful professional framing efforts

While framing devices may enter conceptual modeling indirectly, their significance for guiding innovation should not be downplayed. There existed other efforts for providing conceptual framing to Sakai 3, notably as a socio-technical system straddling technology and real world institutional activities, and as a pedagogical device in support of teaching and learning. In the following I will argue that these framings did not make a notable

impact on the conceptual construction of Sakai 3, because they were not translating their contributions into conceptualizations of the system as a real, evolving material artefact. Even though the efforts were relying on powerful conceptual tools, they did not become represented in the prototyping context, and they were unable to provide a focus to spontaneous modeling efforts in the Sakai 3 design space. The strength of user-centered methods in a context intent on building an actual software system was their reliance on approximations that could convincingly claim a direct connection to the actual artifact. This was most obvious in the case of visual mockups and interactive walkthroughs, which successfully sliced off aspects of the system that could be engaged with without having access to the code.

6.3.1. Investigation project: Glimpses of a socio-technical system

Studies of IT implementations in organizational contexts have repeatedly asserted that activities with information systems do not and cannot stand in isolation from the context of organizational activities within which they are embedded (Kling & Scacchi, 1988, Orlikowski, 1992; 2000). They have also pointed out the informal, tacit and emergent character of work, and contrasted this with the expectation that software will allow for the routinized automation of segments of these activities (Suchman, 1987; Star & Ruhleder, 1996).

The Sakai project was designed to be a software community with an institutional focus and an ambition to engage with its own institutional domain. Many participants were part of instructional support teams at an institution of higher education, and thus they were well positioned to evolve a socio-technical understanding of software technology in the educational context. They were adept at describing scenarios where the

institutional context was entangled with technology, talking about organizational roles, hierarchies, legal requirements, instructional schedules, and many more intricate details of the day-to-day realities of higher education. Some of the participants were also eager to see their own contributions in socio-technical terms, and took the trouble of constructing the conditions of their own epistemic work along these lines.

The Investigation project was a culmination of this approach. It was set up to understand the real, organizationally embedded, practical contexts of instructional work practices around assignments, in view of understanding how a future platform could better support this central collaborative aspect of higher education. The project started with a series of scripted interviews at various universities, which were carefully planned and executed in the above spirit. The presentation of the project talked to these overall goals:

> **"**We plan to sponsor a 3-month investigation phase […] to help us understand the range of people who use Sakai to create, manage, complete, and assess learning activities and how they think about their work. We want to have a solid understanding of the historical issues, user types, and user goals before we begin designing in January. We believe that understanding how various users really think about their work will lead to new ideas for how workflows need to be structured and interrelate to each other. In other words, it should help us to understand the commonality between workflows that currently occur in multiple tools in multiple ways. Also, although we are not immediately integrating with the workflows of communication, scheduling, and grade reporting, we would like to

know how and when users expect their work surrounding learning activities to integrate with those workflows. […]

Therefore, we hope that a broad range of institutions will contribute to this investigation, especially by providing end-user profiles based on local interviews of instructors and students regarding the work they do with tests, quizzes, and other graded assignments.”[67]

The emphasis on workflows and their integration, the variegated contribution of work tools, the acknowledgment of user types and user roles (instructors, students, but also TAs and instructional support personnel), and the goal to understand how users “really think about their work” all talk to the ambitions of the socio-technical tradition.

The interviews were outlined to investigate the socio-technical system around Sakai 2, with a particular emphasis on getting beyond the Sakai community’s bias toward software-centered conceptualizations. The socio-technical focus was also apparent in the details of the interview protocols used by contributors at different institutions for making their interviews. Thus, the introduction of the interview read:

“The main focus of our interview/observation session today is to understand the different activities (e.g., homework, papers, tests, quizzes) that take place within a course, how they are created, disseminated, completed, collected, and evaluated. […]

While you answer questions or guide us through tasks, please focus on the details of how you actually do your work. It may help to think about the last time you performed the task and explain it to us as if we are going to need to perform the task just as you did. Please feel free to be honest and critical even if the way your

work *actually* gets done is not the way you would *ideally* like for it to be done. […]"[68]

Thus, the interviews were focusing on details of work activities as they were actually taking place. Interviewees were also asked to guide the researcher through one of the assignment-related activities that they most regularly undertook, and the protocol repeatedly prompted the researcher to ask for clarifications about unclear details. The interest in the context was also emphasized; interview questions included for example "Could you please describe a typical day for you at work?" [69], and the prompts clearly stated that the activities described did not have to take place in the Sakai platform.

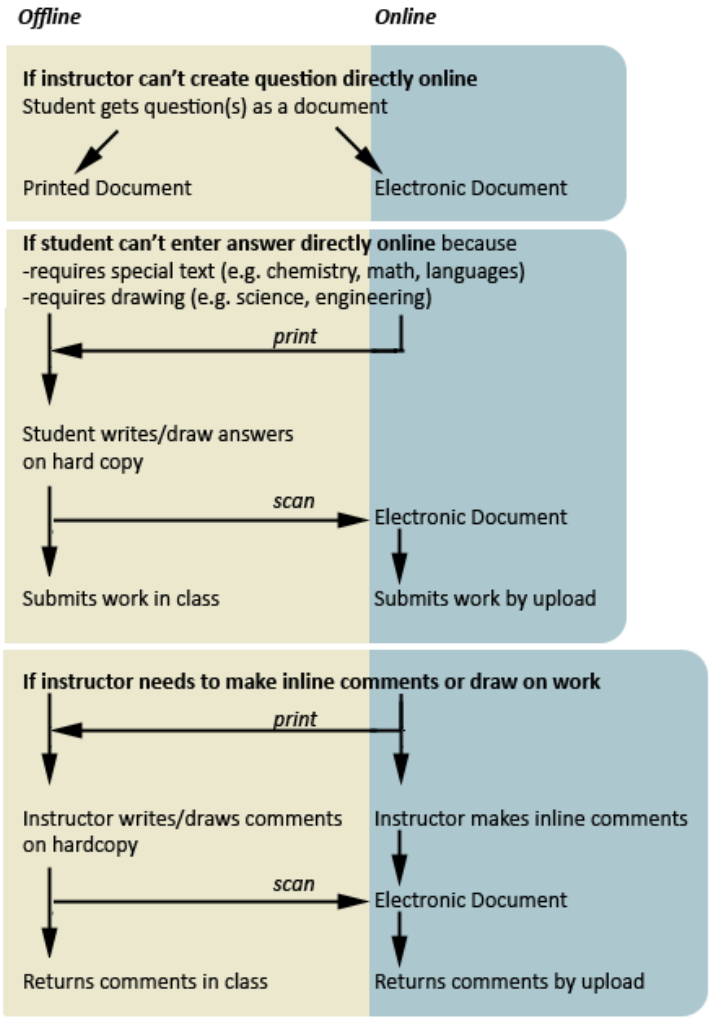
The project itself was a complex and long-term endeavor integrated with the methodology of goal-directed design, one of various user-centered design methods. The interviews were to serve as the foundation for creating personas and scenarios in support of design. Assignments, which constituted the focus of the research, were taken off from the Sakai 3 agenda after the interviews were finished, but despite this shift, personas created on the basis of findings were systematically used in the managed project to illustrate requirements. My focus here is not on the user-centered contributions from Investigations, but their unexpected socio-technical findings which were not taken up in creative design efforts.

The point of failing contribution that I want to make is related to the initial report[70] on the outcome from the interviews. The initial report formulated a set of interesting findings in this respect. It was found, among others, that one of the most harrowing challenges faced by users was not related to how existing tasks were structured within the Sakai 2 system, but the integration of the digital and the non-digital worlds.

The interviews showed that assignments at certain phases of their lifecycle were not digital. The reasons behind the existence of non-digital assignments were numerous: different engineering professions were using visual notation systems and graphical practices which required paper; providing feedback in the immediate context of student responses was best done on physical copies; exams had to be organized in actual physical spaces with supervision, and the provision or supervision of computers would not have been possible. Despite the significance of physical copies, support for the online management of the workflow was strongly desired, so much so that stories about printing, scanning, and possibly even reprinting assignments were common. The research described various online-offline crossing scenarios for content, and included a diagram of reconstructed workflow (Figure 6.5).

In light of the Sakai community's explicit and avowed focus on digital content for Sakai 3, this finding may be rightfully described as unexpected, while very much in line with a socio-technical perspective. Users did not seem to be so much ahead into the world of digital content as Sakai participants envisioned: they were embedded in a world of paper-based content, and they were hoping relief from their digital system in the management rather than the structuring of assignment-related content.

Figure 6.5: Diagram of workflows with digital and paper-based content in the in the initial research report of the Investigation project[71]



The finding was formulated in the conventional format of a research report, which was shared on Confluence, publicized on e-mail lists, and presented at different Sakai events. Despite its wide availability, it never made an appearance beyond the report. What's more, the socio-technical perspective was entirely missing from the web of discourse, and its pervasive efforts of conceptual construction. Thus, in other words, the findings from the research did not spark the envisioning of Sakai 3 as a socio-technical

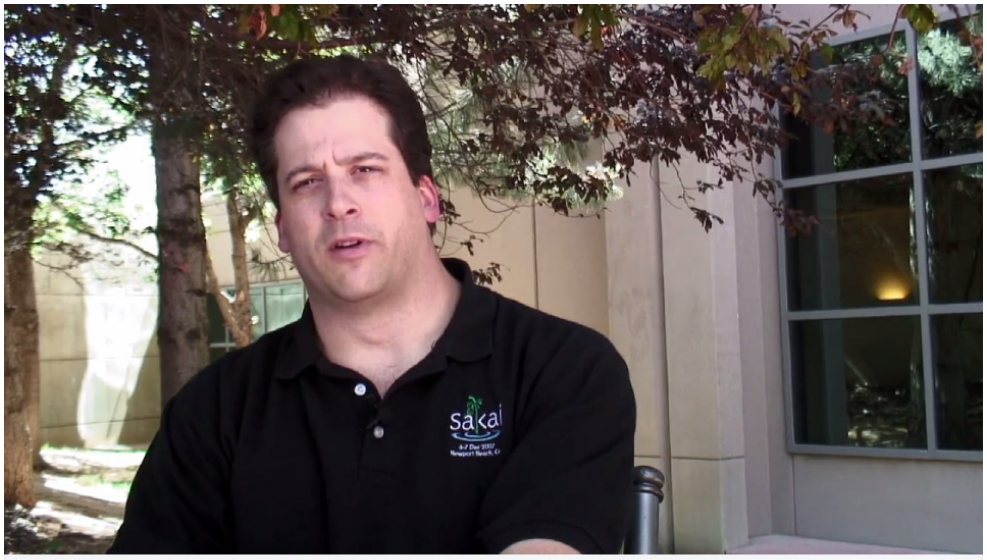
system. Overall, the new platform was not able to expand in a socio-technical conceptual space. Also, the project did not produce a powerful prototype conceptualization, a framing device analogous to the user interface, which could have served the role of focusing modeling efforts. Socio-technical construction was cut short at the stage of sundry empirical findings.

6.3.2. The Instructional Visioning initiative

The second story relates a more systematic effort described by participants as "instructional visioning", which was deliberately and intensively engaging in conceptual construction. The effort also relied on a panoply of framing tools to support its collaborative visioning activities. Like the user-centered and socio-technical approaches, it was also rooted in a professional perspective, that of modern-day pedagogy. A loose pedagogical perspective had been cultivated by a lively sub-community within Sakai, calling itself the Teaching and Learning (Discussion) Group (T&L Group). This group, and the instructional visioning initiative within it, created an eclectic epistemic focus from a pedagogical interest in teaching and learning, a practical interest in instruction in higher education, and a technical focus on software tools. The approach here was different from user-centered or goal-directed design in that it was not relying on established methodologies. Instead, participants were attempting to invent their own way around the uncharted territory of domain-driven innovation in education.

The instructional visioning initiative launched a Google spreadsheet for the collaborative collection of simple, technology-agnostic learning capabilities, and their possible technical implementations. More than 25 people contributed to the spreadsheet[72], which grew to contain more than 160 different capabilities. The capabilities were later

arranged in themes or facets, evolving into a synoptic diagram called Design Lenses, which was widely cited in Sakai as a valuable conceptual framework for understanding what Sakai 3 will be. The community momentum of the initiative was significant within the history of Sakai, and it also had ambitious goals for domain-driven design, as described by Josh Baron in his retrospective account of the initiative, which was recorded on video after the publication of the Design Lenses (the text is my transcript of the recording):



"The process got started at last year's conference in Boston, the 2009 conference, where Sakai 3, kind of the next generation of this technology, was starting to be visioned and discussed. And as those conversations were taking place, many of us began to feel within the teaching and learning community that there was a real role for us in helping to define the learning capabilities that Sakai 3 should support, not only for our more traditional instructors who are on our campuses, but I think even more importantly, for those innovators, for those early adopters

faculty, who are doing very new and very powerful things with technology in terms of teaching and learning.

And so after the Boston conference, a number of us, both those who have been in the Teaching and Learning community for a couple of years, and new folks, including rather excitingly leaders from the Open Source Portfolio Group within Sakai also came together to try to solve this puzzle of how do we really work with designers and developers to influence the design of Sakai 3 as it emerged. And after a very kind of lengthy, organic process of discussing how we go about doing this, this concept of learning capabilities emerged, where rather than trying to put forward a set of requirements, a set of technologies or functions that we wanted Sakai 3 to be able to support, we decided that looking at this more from a user-centered perspective was going to be more effective for designers and developers. And so we began by simply listing out in a Google spreadsheet those learning capabilities that we thought were important in Sakai 3. So these were things like the ability to engage in class discussions. So again, avoiding specific technical or functional terminology. Very quickly, through the work of many-many folks within the community we had 160 documented in a Google spreadsheet, which was great, but at the same time a bit chaotic, and not terribly useful to have this raw list of a 160 capabilities. So from that, we began to cluster these, and look for themes that these would kind of fall under, so originally about ten or eleven themes did emerge. Then we began to categorize these learning capabilities under these themes. I will kind of skip a very significant effort that the community engaged in to refine those, it took many-many hours on phone calls and in

Confluence to figure out the right final definition of these themes. And from that work emerged this concept of Design Lenses. So at this point we have now defined seven design lenses, under which now all of these learning capabilities kind of fit, and the reason is we call them lenses is that our idea, that is still an idea at this stage, [our idea is] that as Sakai 3 is being designed and developed, those working on that will be able to view their work through the eye of teaching and learning folks by peering through these lenses. So one of the lenses is openness, which I think is a value that our community embraces on many levels, both open source code, as well as open educational resources. And so a designer might be looking at a discussion capability, and how to design that, and they might peer at it through this openness lens, and from doing that realize, that someone might want to take a discussion that initially was closed within Sakai, and open it to the rest of the world, and others could come in and participate in the discussion, or take the discussion that took place and push it out to a blog, or a social networking site, like Ning, again engaged the broader community outside of the course in the discussion.

We have got tremendous amount of very positive feedback from the community, and a lot of also great ideas about what we have to do next. […]"[73]

From Josh Baron's account it is clear that the instructional visioning initiative was an attempt in creating a framework for plugging the instructional perspective into the process of technological innovation. The originators were seeking an approach that would put instruction in the position to drive innovation, and the suggested solution was to base new technology on innovative teaching practices. From the beginning, this domain-driven

contribution was also conceived as a participatory effort, which would involve non-developers within the Sakai community and beyond, among the instructors using the platform.

The participatory, inclusive approach appears to have been perceived as a major challenge insofar as the instructional perspective was understood to be by essence non-technical. As an early formulation of instructional visioning stated:

> "These would be stories that would capture, at a high level, innovative teaching and learning strategies (grounded in teaching and learning theory/research) that could be facilitated by Sakai. They would be tool/technology agnostic "pie-in-the-sky" narratives who's [whose] goal would be to provide a "functional target" at which future development could be aimed. The ultimate goal would be to get teaching, learning, and research needs out ahead of development efforts."[74]

How can technology-agnostic understandings about teaching and learning truly become drivers in innovation with technology? Early discussions, such as the above, were seeking to pin down a genre which could express innovation within teaching and learning, serve as a bridge to technical development, and also be easy for Sakai participants to understand.

Figure 6.6: Headers and a selection of the examples in the original excel file shared by David Goodrum[75]

Trad T&L Task	Alt T&L Task	Sakai 2	Capability	Capability+	MCS [multi-section course] example	PITS example
I need to meet with my students somewhere	We need a place to work together	Course site	Site creation	provisioning; site templates	Multi-section course with both common space and section specific space	Construct a virtual world in three dimensional space
I need to see who my students are	We need to see who's participating	Roster	Add/Delete users	daily batch SIS integration	dynamic, real-time SIS integration	
I need to know about my students	We need to introduce ourselves to each other	Profile	Show name, photo ID, major, interests	SIS data, demographics, personal information	Different people permitted to see varying amounts of institutional and personal information	Information as well as professional and personal connections, presence, availability, current location

The spreadsheet structure created by David Goodrum became successful within this space. He was suggesting "a rough attempt at starting with a basic T&L need, and then expand that to describe functionality as well as complex and pie-in-the-sky needs." His spreadsheet (see Figure 6.6) started with descriptions of traditional teaching and learning tasks in everyday language (with an additional column for alternative descriptions), alongside the corresponding Sakai 2 tools, and the description of the capabilities within these tools. Each task could be further extended into descriptions of innovative, but technology-agnostic "pie-in-the sky" solutions. Besides the headers, almost two dozen examples were also provided.

The spreadsheet was an innovative approach for creating a primacy for the instructional perspective insofar as it acknowledged the preliminary existence of software technology in the design space. To do this, it outlined a specially constructed vantage point, where some of the future could be reached by means of excavating essential instructional motivations buried within the existing Sakai 2 platform. The approach also made space for the inclusion of instructional tasks which had not been previously addressed in software technology.

As the number of listed learning capabilities grew, the document became more and more unwieldy to process. The group was thinking to move from collection to validation, but the list did not provide clues for internal comparison and coherence across the items. It also became clear that the output could not be shared with a broader audience in this raw form, as Josh Baron's account suggested. Arranging the individual capabilities under broader themes appeared a logical next move, but as it happened, this also redefined the work, lending it a comprehensive and normative edge. The Design

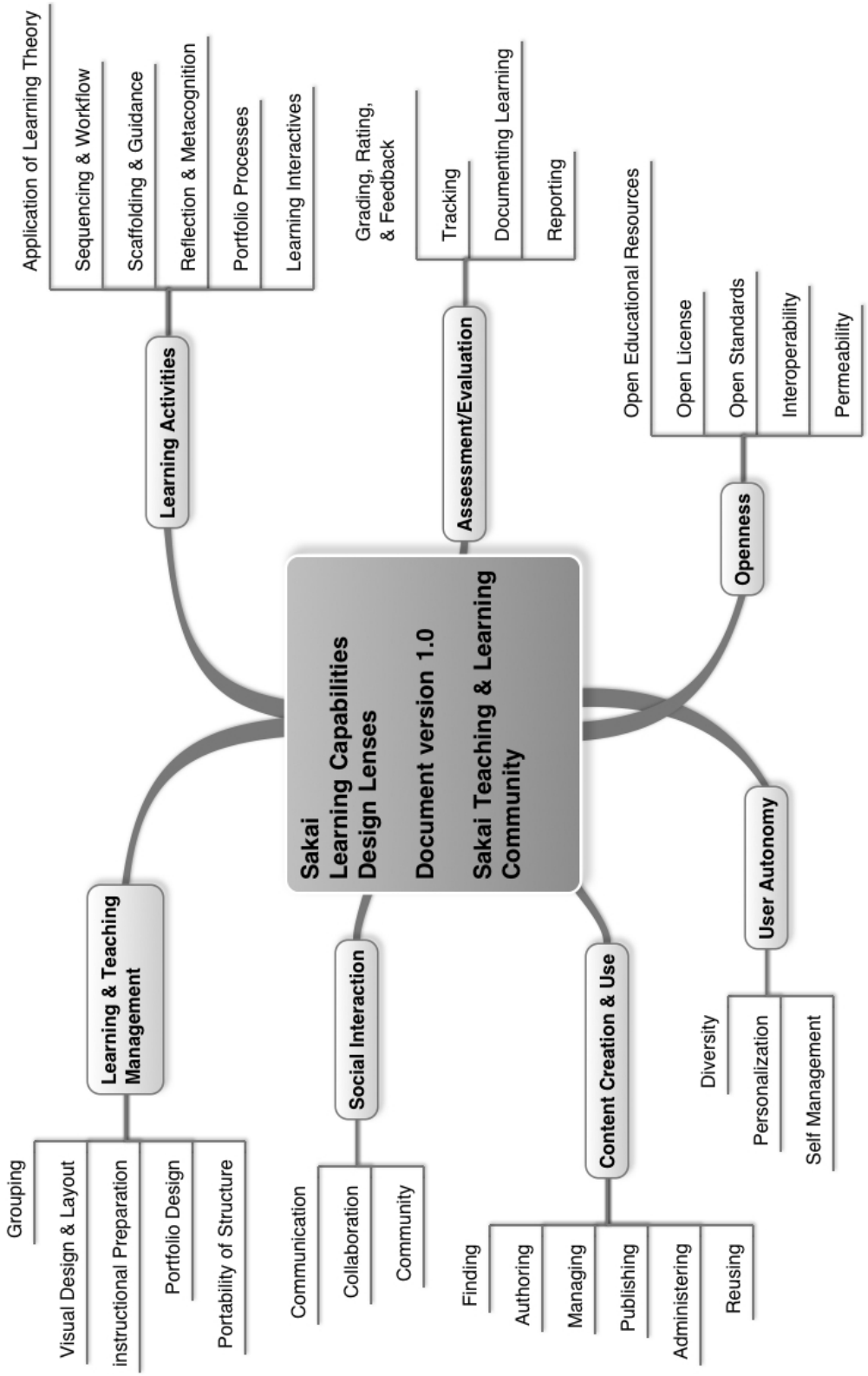
Lenses diagram (see Figure 6.7) emerged as a comprehensive conceptual framework which should not be bypassed in the design of a new platform. As its suggested use was described:

> "Each Design Lens is not an area of functionality, but rather a perspective from which the entire system should be considered.
>
> Facets are defined in general and also in relation to desired outcomes in Sakai. All lenses and facets need to be considered when working on any new functionality in Sakai."[76]

Broad participation in the making of the document also underlined its comprehensive normative character:

> "The Lenses and Facets aim to capture not just what teachers and learners need or want today in terms of learning capabilities but what we as a community believe they will need and want in the future."[77]

Figure 6.7: The Design Lenses[78]



The above accounts described a strong framing, which was further reinforced by the name of Design Lenses. The name suggested that the themes were providing an external, authentic vantage point from which technology could be contemplated. This vantage point could be applied to scrutinize technology in terms of its alignment and responsiveness to the needs and values of its context. In the concrete situation of Sakai 3 moving further along toward completion, this framing was intended to facilitate the belated integration of the instructional perspective with the process of innovation. As Josh Baron suggested, applying the lens of openness would not only allow finding areas where the system was lacking, it could also serve to augment what was already there.

The Sakai community appeared to be willing to play along in the suggested application of the Design Lenses. The diagram and the surrounding effort received a preeminent place in various venues of the Foundation, and it was presented as having significant promise and potential.[79] Within the Teaching and Learning Group, the T&L Design Lenses Group[80] was formed in the summer of 2010, around the same time that the managed project came into existence. The group included two dozen core contributors, set up its own Confluence space, email list and regular online discussions. Participants set for themselves the task of taking the Design Lenses work forward in Sakai 3. As the group was seeking to define the nature of their contributions, they became taken up in the Minispec effort which had been recently initiated for managing the design of the new platform. The idea behind the Minispec effort was to document requirements in support of UI design. The following email provides a glimpse of the circumstances of the involvement:

> "ACTION ITEM: Based on feedback from Clay and others involved in the Sakai 3 Project, the group decided that it would be useful to spend time over the week reviewing and commenting on the current set of minispecs. Feedback could cover a range of topics including learning capabilities that are not addressed in the minispecs, formatting of the minispec or suggestions/examples of how the lenses could be applied to them."[81]

The Design Lenses group became involved in the review of minispecs, but the actual Design Lenses document did not have a specific role to play. I am not aware of any substantial, formative contributions that were explicitly linked to the lenses or showed the influence of their perspectives in an indirect manner. More importantly, there was no evidence of related conceptual construction efforts in the web of discourse.

Thus, the Instructional visioning effort created conceptual tools with the explicit purpose of making way for the pedagogical perspective in contributing to innovation with Sakai 3, but its attempts were not successful. Instead of suggesting a coherent model of the operation of the software platform, the Design Lenses was showing it in fragmented, disconnected facets.

6.4. Discussion

6.4.1. The role of prototypes and prototyping in framing the direction of design

The case study in this chapter attempted to provide an outline for conceptualizing the contribution of professions to the direction of innovation. I was suggesting that the influence of UX-oriented professional practices was based on the continuity that the practice of prototyping created around the development process, and the coherence that the actual prototypes were able to project for the evolving platform.

The various design professions, including user-centered approaches to software design, have been well known to rely on approximations of the artifact, including sketches, plans and prototypes (Cross, 2000; Römer, Pache, Weißhahn, Lindemann, Hacker, 2001). It has also been generally acknowledged that these external representations are central in the exploratory conceptual processes that characterize design (Beaudouin-Lafon & Mackay, 2007), and as such, they can be sseen as part of a distributed cognitive process, but the nature and dynamics of their contribution has not been widely studied. In an *in vivo* observational study of mental simulation practices in design, Christensen and Schunn (2008) looked at the role of two types of external representations, sketches and prototypes. They found that sketching commonly co-occured with mental simulations, but the engagement with prototypes reduced the number of mental simulations. On the basis of this finding, they suggested that

> "3D external representational systems provide the option of using alternative strategies for reducing uncertainty, thereby limiting the need for running mental simulations" (p. 16.)

I have similarly found that prototypes were not directly implicated in mental simulations. The pervasive practice of the design review suggests that in the distributed context, prototypes are intended to be devices that facilitate communication about design rather than direct engagement in construction. My case study of Sakai 3 further suggests that prototypes were associated with the activity of prototyping. It is in connection with this practice that they acted as a framing for conceptual design, and their role in reducing uncertainty in communication can be interpreted with respect to this broader context. I have shown that prototypes associated with the user experience perspective were

suggestive of a powerful conceptual coherence or framing of what was being built: in their conceptual modeling efforts, participants were grappling with the meaningful construction of a user interaction system, particularly as it was channeled to them by the specific user interface prototypes that became available within the design space. These guiding representations of the interface were able to act as powerful framing devices even when participants were not directly engaged with their manipulation. These artifacts made possible a distribution of design where the routine activities of mockup production coexisted with creative efforts at reimagining the interface and the broader system of user interactions. For framing devices to be able to operate in this manner, it was significant that they were prototypes of the evolving artifact. On my analysis of the Sakai 3 case, prototypes projected coherence for the future platform in terms of the user interface, and conceptual modeling was seeking to recreate this coherence in its sense-making efforts.

I would like to argue that these framing inputs are important for grasping the character of the artifact which has resulted from the expansive process of innovation. I have described the product of the design process of conceptual modeling as a widget-based dashboard interface for higher education, which places online group activity in the focus of user interaction with the software platform. While it is straightforward to grasp the novelty of Sakai 3 in terms of these specifics, accounting for the system in generic terms requires that we rely on its origins within the design space. In this light, Sakai 3 provides a novel logic of user experience for the higher education context, which can be traced back to the framing of the design process.

6.4.2. Missing perspectives

Besides the user experience perspective, other potential framings were outlined within the Sakai community. The Investigation project was seeking to grasp Sakai 3 as a socio-technical system, and the Instructional visioning effort attempted to envision an instructional platform, but neither of these perspectives was able to influence the direction of design. Most importantly, the socio-technical and pedagogical perspectives were not apparent in the efforts of conceptual modeling that sprung up in the web of discourse. It is with respect to this lack of influence on conceptual expansion that we may say that Sakai 3 was not designed as a novel socio-technical system or a novel pedagogical platform, and its novelty cannot be accounted for in these terms.

While the socio-technical and pedagogical perspectives did actually appear in systematic efforts to shape the design of the Sakai platform, other significant facets of software's contribution to the patterning and organization of the human activities were blatantly absent, or received only sporadic attention. Discussing the significance of software code in the regulation of human life, Lessig has for instance suggested that the forces that constrain human action may be grouped under four broad areas: the law, social norms, the market and technical structure (1999). When the development process of Sakai is examined under the lens of Lessig's theory of regulation, we see that significant facets of the social embedding of the platform remained hidden and unexplored.

The legal ramifications of the system were visited by the Sakai community in a sporadic manner. The US regulation of private educational data in FERPA (Family Educational Rights and Privacy Act) was mentioned from time to time in discussion, and

it became clear that the different US-based institutions were working with different local interpretations of the law within the online context. Meanwhile, there was no effort to systematically review the application of FERPA at the different educational institutions, or to proactively formulate guidelines to be followed in the online context. Also, with respect to the international character of the development effort, there was no interest in charting privacy practices across the different countries represented in the Sakai 3 effort, and within the broader Sakai community. Thus, it may be said that the legal lens of data protection and privacy did not enter into the design of the Sakai platform.

Considerations about the market of educational platforms were central in the initial creation of an open source community for developing educational software, but after this initial period they were rarely addressed. It has been argued that open source itself may be seen as a reformulation of a market-based logic of circulation, which replaces money with alternative forms of motivation in the exchange of goods and services (Benkler 2002, Raymond 1999). Related to this I have shown how the open source approach may be understood as an effort to create an epistemic space for software development. This involves the fostering of immediate developer buy-in and considerations for long-term maintenance or development in the design of the artifact, notably in connection with modularity. These elements also appeared in the development process of Sakai 3. At the same time, parallel considerations of user buy-in and long-term evolution of user interest did not play into the design of the platform.

With respect to the area of social norms it may be said that the design of Sakai followed a conservative approach, which was seeking to fit the platform with existing institutional and social norms, rather than attempting to shape existing norms through

software. In this respect, it was untouched by a broad area of critical innovation in education, which seeks to address the throes of education with innovative technologies. Participants in particular had little interest in the various flavors of online education, and its repeatedly heralded potential to address the soaring costs of education, or global disparities in the cost of education (DeMillo, 2011). Similarly, I did not come across any mentions of recent initiatives of using the web for opening up educational resources in the online context (Atkins, Brown & Hammond, 2007), despite the fact that at least one of the participating institutions (Stanford) had been active in this field.

6.4.3. The framing role of professions

My analysis of the role of UX-oriented professional practices in the design of Sakai 3 suggests that professional practices may be understood to be instituting systematic selections over knowledge which frame the space wherein conceptual expansion will take place. In the above, I have been talking about user interface and user experience in general terms, without attempting an explanation of the particular selections these imply for a non-technical, human-centered framing of the artifact. My analysis of unsuccessful framings has also suggested that these selections could be different: Sakai 3 could have been alternatively framed as a socio-technical or a pedagogical software system. I will address these questions in the next chapter, where my goal will be to outline how this professional framing guided investigations about the real life contexts where the system was to be used.

Taking this account further, professional practices may be understood as general strategies which are to be applied within a design space and which bring about systematic

selections of knowledge for projecting a coherent conceptual account of the envisioned artifact. In the concrete case, the user experience approaches were framing the new software platform as user interactions with a user interface. The analysis of the concrete case can provide insights for constructing alternative strategies, which create different framings for envisioning the system. As part of a professional framing strategy, prototypes may be understood to provide experiential representations of the conceptual coherence of the system. It is in connection with this facet that they can be suggestive of a conceptual framing in the context of conceptual modeling and innovation.

I will return to the problem of human-centered framings in the next chapter, in connection with an account of the failure of Sakai 3, which points to the possibility of the conceptualization of the artifact as a social system, with processes of information-based interaction across many users. I will argue that human-centered framings are in need of framing devices similar to those applied in user experience design. I will revisit the analysis of this chapter to suggest ways in which an innovative design practice around this alternative perspective may be constructed around conceptual models of the artifact, which may act as framing devices in connection with the activity of prototyping.

DESIGNING THE SAKAI OPEN ACADEMIC ENVIRONMENT: A DISTRIBUTED COGNITION ACCOUNT OF THE DESIGN OF A LARGE SCALE SOFTWARE SYSTEM

VOLUME 2

A Dissertation
Presented to
The Academic Faculty

By

Klara Benda

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Human-Centered Computing in the
College of Computing, School of Interactive Computing

Georgia Institute of Technology

August 2014

CHAPTER 7. THE SOCIAL-TECHNICAL GAP

7.1. Introduction

The importance of making space for socially grounded imagination in the design of new software has been pointed out within Human-Centered Computing, notably by Bannon (2011) and Dourish & Bell (2012). Both writings contrasted the need and possibility of introducing social imagination into the current, technically oriented software design process. At the same time, these authors have not provided an account of the thought processes that the contribution of social imagination may involve. In this chapter, I will attempt to formulate what a social approach to the design of software may be like. Unlike in the previous chapters, the discussion will emerge from an analysis of the shortcomings of the design process. The new Sakai platform was piloted twice, and both pilots highlighted inadequacies of the information model of the back-end to anticipate patterns of use in the institutional context of education. In both cases, an understanding of the nature of the challenges emerged during the pilot, and resulted in the redesign of the back-end. While participants were able to connect technical structures with social patterns of usage once the platform came to be used, these connections were not explored during design. The process of design was itself relying on the separation of user-facing design from technical back-end design, and the latter was characterized by a tinkering approach, with a focus on making software components work together. In the discussion, I will suggest that a social-technical approach to software design is also possible, and it may come about by placing the epistemic stance of sociology within the cognitive-

epistemic context of the processes of software design. For the latter, I will rely on my earlier analysis of conceptual modeling and framing devices.

7.2. Case study: the lack of social perspectives in design

7.2.1. The failures of Sakai

As Sakai 3 was coming alive in institutional pilots, first with the Sakai OAE Q1 release in October 2011, and subsequently with the first official release, Sakai OAE 1.0 in September 2012, it was showing alarming signs of inadequacy. The difficulties were decoded as problems of performance, which pointed to the inadequacy of kernel components under the load patterns of real world usage: in sum, the new kernel did not withstand the test of the real world. For both releases, the problems resulted in a reconsideration of component choices, and their replacement by new kernel components. The lead of the kernel team described the problem for the Q1 release as follows:

> "So the Q1 release is out the way (in 4h) and we are moving onto working on Q2. […] This is all *with hindsight*, in the cold light of day, and brutally honest. / The Q1 release was (i[m]vho [in my very humble opinion]) a disaster. We were targeting to be able to support 4K [4000] users on a single JVM [Java Virtual Machine, i.e. a single server, practically] and we struggled to support 30." [82]

The lead subsequently traced the issue to the choice of a content management platform designed for Enterprise Content Management, which is characterized by top-down content publishing, but unsuitable for the Social Media use case of Sakai 3, where everybody contributed. The platform was also not designed to cater for the complex permission patterns that social media required in the world of higher education. The new content management solution did not fare better: it was struggling to serve up pages made

up of hundreds and hundreds of small pieces of content, which were often retrieved piecemeal with so many search queries and requiring a lengthy parsing of permission settings. It also faltered when a high number of users were simultaneously working on the same content page: the platform performed only a portion of the updates that were made as a result of a lack of appropriate conflict resolution mechanisms for these types of situations. The problematic patterns of usage were not foreseen by the developers, but they turned out to be common in the world of education, where cohorts of people live at the rhythm of deadlines.

Why were these challenges not known in advance? Why was it that understanding the logic and the pattern of activities in conjunction with the educational software platform had to take the costly route of a real world pilot, instead of the path of preliminary design and modeling? This is the problem that I will be exploring in the case study of this chapter.

The performance problems were interpreted by many in the Sakai community as the failure of S/OAE. Various commentators suggested that technical problems had played a significant role in the withdrawal of five of the seven universities in the summer of 2012 (see the Timeline of events in Figure 7.1) [83], which resulted in the collapse of the S/OAE managed project, leaving Georgia Tech and Cambridge as sole institutional partners behind S/OAE. The reasons leading up to this collapse – what was discussed within Sakai as the why of S/OAE's failure – were certainly more complex, and included an array of technical and institutional factors[84]:

In the words of David Ackerman, Sakai Board Chair at the time, the reasons were:

"1. OAE does not scale yet

2. OAE does not have all the features of an LMS [Learning Management System]

3. 1. and 2. are taking too long"[85]

Nico Matthijs UX lead summarized the "key reasons cited for these departures" as:

"- Concerns about the scalability and viability of the Nakamura architecture

- Slower than desired progress on feature development

- Misalignments between stakeholders"[86]

Thus, the first release of Sakai OAE was perceived to be far from the expected set of functionalities, notably it did not include any educational functionality, and the educational customization of its generic social media capabilities was also partial. It was also afflicted by performance bottlenecks, which challenged the viability of the technical choices behind its foundations. While some may have come to doubt at this point the possibility of ever achieving the original goals, the concern that institutional partners actually voiced was about the practical achievement of their goals within a reasonable timeframe. The wavering of expectations and hopes was pointed out to be confounded by dire financial circumstances afflicting some American universities around this time[87], as well as the pressures of local schedules of technology adoption (both NYU and Charles Sturt had made strategic decisions to replace their seriously inadequate Sakai 2 installations within a short timeframe).

Figure 7.1: Timeline of back-end development

2005	
March	Exploration of migrating Sakai 2 to JSR-170.
June	JSR-170 final release.
2006	
2007	
July	First report of implementing JSR-170 Sakai 2.
November	Release of Jackrabbit as a Contrib tool in Sakai 2.
2008	
August	The sakai-kernel email list is created.
early December	The APOC kernel solution is selected over Sling and OSGi.
December 15	Sakai 3 – A Proposal
late December	K2 team is formalized as an open source team.
2009	
April	The return of Sling and OSGi.
2010	
October 7	Performance issues aired by Ian Boston.
early October	Experiments with Sparse Map started.
October 18	Sakai OAE Q1 pre-release
December	Experimentation with Solr started.
2011	
February	Sparse Map is officially merged into K2.
September 8	Sakai OAE 1.0
late October	Collection of Performance Issues in NYU's pilot is started.
2012	
January	OAE architectural review with OmniTI consultancy.
Summer	Withdrawal of Sakai OAE project partners.
October	Outline of Sakai OAE's new architecture is announced.

While the project itself dissolved, S/OAE has continued to evolve, and lives on today under the Apereo Foundation, a new umbrella organization that houses educational open source initiatives, including the Sakai Foundation. Whether S/OAE has failed and to what extent remains a matter of perspective. It is not my goal to address this question, or to trace the possible causes of the different shades of failure that have been ascribed to S/OAE. Meanwhile, the two episodes prompted soul-searching among direct contributors, which, together with the history of engagement with the technologies in question, provide insight into the ways of knowing that accompanied the making of K2, and most importantly, the ways of knowing which emerged as legitimate and valuable in connection with the pilots, but had been conspicuously absent in earlier phases.

In my account below, I will argue that the development of the kernel was afflicted with a lack of social perspectives, and the lack of a social-technical imagination which could have unfolded from a social outlook. This prevented participants from envisioning the working of low-level platform technologies as socio-technical: as technologies which function in conjunction with a specific and knowable social context. Besides the lack of socio-technically grounded thought experimentation, I will point out the lack of socio-technical framing devices, which could have guided conceptual modeling efforts in the same way that prototypes of the user interface have been shown to do in case of the UX-perspective. I have to warn in advance that this argument is hypothetical in the sense that it is based on speculations about what could have been, supported by insights from parallel case studies described previously. Also, I do not want to claim that engagement with the social would have prevented the specific failure of S/OAE. Engaging with the social could have still meant that the relevant aspects of activities remain overlooked.

What I do suggest, however, is that the kernel struggle highlighted an epistemic gap in software design, and a pattern wherein the social was known after the fact of implementation, rather than before. This gap makes experiments in social software potentially costly and fraught with unintended consequences of technical choices. I am not describing better ways of knowing which result in better outcomes – only different ways of knowing, which result in different outcomes. Overall, this difference involves increased reflexivity within an epistemic domain. This can be applied to a more cautionary approach, but it also encompasses views of the world and how it is made to appear within the design.

7.2.2. The origins of the kernel and the kernel team

Unlike previous chapters, this case study will focus on the low-level platform technologies behind Sakai 3 which came to be known as a new kernel (K2), subsequently labeled Nakamura. The creation of a bundle of server software and its separation from user-facing software development approaches was originally undertaken for Sakai 2; the outcome of this effort was the Kernel. Intensive engagement with these low-level technologies created momentum for exploring better options, which eventually resulted in plans for a complete rewrite of the Kernel to support the new Sakai. K2 was built around a technical choice that had been incorporated in K1, the JSR-170 content repository specification, and its implementation in the Jackrabbit (JCR) content management platform. The first S/OAE release, Q1 incorporated further open source modules supporting the JSR implementation, notably the Apache Sling framework, which supports RESTful HTTP-based access to a content repository. Jackrabbit proved to be a wrong choice at the time the Q1 was released in 2010 October, and it was soon replaced

by a homebrew content management library, SparseMap, supplemented by Apache's Solr open-source search library. The case study will revolve around Jackrabbit, Sling, SparseMap and Solr, and the practices of engaging with these technologies. For situating and describing these technologies, I will rely on participants' accounts. Local accounts have proven to be sufficient to give a sense of their contribution and place within Sakai 3 for the purpose of the case study.

After the first official release of S/OAE, the kernel was again redesigned with support from a performance consultancy (OmniTI). Apereo OAE has since then swapped earlier Java-based server technologies for Node.js (a server-side Javascript approach), as well as a reliance on distributed architecture with various virtual machines. The design of this latest architecture is beyond the goals of the present analysis.

The separation of the kernel resulted in the creation of a kernel team. Starting with Sakai 2's kernel development, the efforts were spearheaded by Ian Boston, who was later appointed lead architect of Sakai 3. Initially, K2 had four contributors committing code on a regular basis. As the overall Sakai 3 effort gained momentum and attention, the kernel team also grew in size, but did not go beyond a dozen participants. When the managed project was outlined in 2011, Ian Boston, the engine of kernel efforts, voiced his disagreement with the curtailment of open-source practices, and withdrew from the project. Meanwhile, he continued to be an active contributor until the demise of K2. The managed project brought another change in terms of a series of small cross-functional teams including developers from both the UX and kernel side. This reorganization was intended to remedy the disconnect of the kernel from the user-facing, functional approach.

7.2.3. The introduction and failure of Jackrabbit

K2 was the legacy of efforts to redesign what did not work in K1 underlying Sakai 2. Sakai 2's problem was that file storage mirrored the structure of the user interface: whatever files belonged together from the perspective of the user, like a lesson plan with materials, came to be stored together, as files within a folder. This meant that every time a file was inserted into a new context, represented by the user interface, it had to be copied into the folder associated with the new context. JCR was an open-source platform which brought the promise of solving this problem by creating an abstract layer of file storage and an API with universal identifiers to stored pieces of content. JCR allowed for creating references to the same digital resource from various locations and contexts in the user interface without the need of duplication. I will describe JCR's adoption at length in Chapter 7, here I will only provide a short summary of the process. Most importantly, JCR was first implemented within the Sakai platform by Ian Boston in 2006, shortly after its first release, as a proof of concept type of prototype. This early implementation grew into a full-blown alternative file-storage support within the Kernel of Sakai 2 in 2008. It was made available as a contrib-type library in official releases from Sakai 2.4 onwards. Despite the successful integration, the perception was that the potential of the content-management model in JSR-170 was curtailed by the limitations of Sakai 2's software ecology. This general discontent led to the exploration of alternative server-side platforms, which in turn led the creation of the sakai-kernel email list, and eventually to the proposal for a complete server rewrite. JCR was the fix point within this process, around which further technical choices came to be organized. In other words, K2 inherited JCR from an earlier process of selection, driven by the discontent with Sakai 2

and a related prototyping effort by Ian Boston. Then, in the performance crisis of the first release of S/OAE, JCR and its underlying architectural model was found to be the main culprit behind the troubles.

The reconsideration of JCR followed after a several week long performance crisis, which affected the upcoming release. As common in software development, the release was already expected to be slightly overdue. It was around the day of its originally planned release that severe performance issues were discovered. Opinions were divided about the severity of the problem, and the possibility of saving the release. The director of QA (Sakai's pre-release testing) wanted to ban the release, while most members of the server team were of the opinion that the software may get slow with time, but that is normal for a pilot release, and should not interfere with putting available functionality to a test in the impending pilot. It was also suggested that the release could easily be followed by a patch. In the heated atmosphere of these discussions, Ian Boston came to the conclusion that Jackrabbit's architectural composition was inappropriate to Sakai's goals, and the performance on these architectural foundations was beyond repair:

> "Unless someone has a miracle up their sleeve this is about as good as it's going to get for the core server until JR 2.2 [an expected new version of Jackrabbit] is released." [88]

His account of the problem was shared on the day of the release:

> "So the Q1 release is out the way (in 4h) and we are moving onto working on Q2. […] This is all with hindsight, in the cold light of day, and brutally honest.
>
> The Q1 release was (i[m]vho) a disaster. We were targeting to be able to support 4K users on a single JVM and we struggled to support 30. Most of [the] server

was Ok but sparse and complex searches were diabolical as was anything that tried to write while other things were reading. Reading performance was/is fantastic but our target audience have write permission and so the server is not 100% read."[89]

The early account was followed by an analysis identifying the architecture of Jackrabbit as the underlying cause of troubles:

"The Problem.

In the Q1 release we hit a series of problems with slow performance […]. This highlighted a number of things that I think we just have to accept.

1. Our use cases are not a good match for those of Enterprise Content Management where most users are readers, as they are far more aligned with those of Social Media where everyone is an editor.

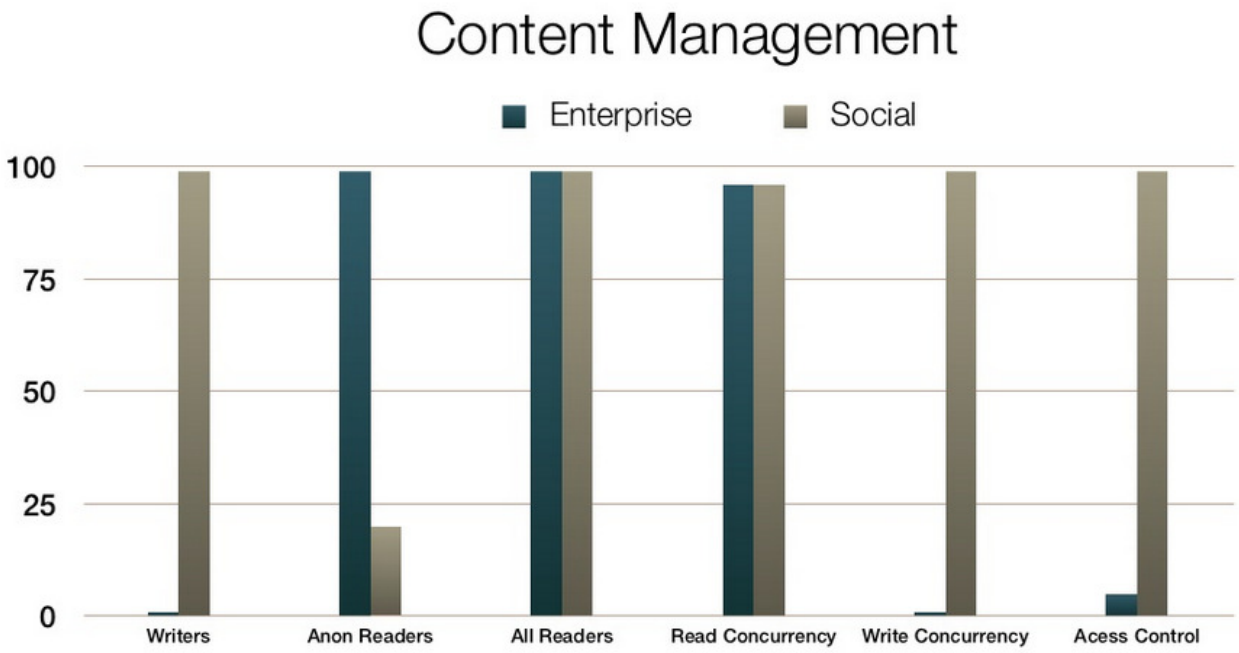
2. The requirements coming through from the UI are not a good match for David's model and Jackrabbit as we frequently have millions of child items, and sparse permissions sets.

Those two top level issues created significant problems inside the jackrabbit code base that resulted in it behaving in ways that it was not expected to. The highly efficient read caches, that make JR so good in [the] ECM [Enterprise Content Management] space and give it most of its performance were frequently flushed by queries resulting in high level of inter thread blocking as more items were read from the persistent store. The ACL structure and layout required by some of the Social Media type use cases causes most of the Access Control Systems to bypass

> the highly optimised pathways, resulting in more synchronisation and blocking between threads. Those were just two of the problems encountered."[90]

Basically, Ian Boston outlined that the content model of the Jackrabbit system was designed to do one thing, and the user facing functionality of Sakai 3 was designed to do something different. This difference had two facets which interfered with performance. The first one was related to basic patterns of read and write usage: Jackrabbit was optimized for a type of situation with a centralized model of publication, referred to as Enterprise Content Management. In this pattern, there is a small group of publishers who publish the content, and there is a large group of readers who read it; publication may result in high interest on behalf of the readers, which means a potential for high reading loads for specific pieces of content. This pattern was later illustrated at a conference presentation with the telling graphic reproduced in Figure 7.2.

Figure 7.2: Comparison of read and write requests in Enterprise and Social Media type of web-based content platforms[91]



The second facet of the difference was related to the interplay of the internal organization of content units and the permissions added to these units. JCR was designed to serve a structure of content which was described after David Nueschler, its original author, as David's model. David's model expects content to be arranged in one or several tree hierarchies, where units of content are densely connected, and permissions are aligned with the hierarchy. Ian Boston suggested that parent units of content in Sakai had a lot more children, without many layers of depth. Lacking more information I speculate that he referred to situations like the one where a class of a hundred or more students contributed one single submission per student. More important than the exact nature of the difference in the content hierarchy was the distribution of permissions within Sakai, which were zig-zagging across the content structures (in Ian Boston's terminology they

were sparse). JCR's assumptions about well-aligned permissions were adopted as a default, but they had to be overwritten by custom code in all other cases. Ian Boston suggested that this custom code was highly inefficient.

7.2.4. Participants explain the failure to themselves

The realization of the inappropriateness of JCR prompted some soul-searching within the server development team, with participants asking themselves why the problem had not been noticed earlier. There were three types of answers to this: (1.) the lack of preliminary load testing; (2.) the UX-driven approach, which meant that requirements were not known ahead of time, and many architecturally significant features were revealed to the server team shortly before the release, so they could not be designed for; (3.) and finally a waterfall development model with a unidirectional flow of requirements, which placed the server team in a position to fulfill requests coming from UX-designers, without instituting sanity checks on what could be achieved, and eventually encouraging a feature bloat. The following discussion excerpts illustrate these different response types:

1. Lack of load testing:

   "As an aside, Why didn't we find this out sooner? / We have known for some time that there are contention issues [i.e. issues of resource contention for access to a shared resource], but not had any load test evidence to support that, and no resource to focus on contention over features." [92]

2. One way flow of requirements in a waterfall development model:

   "We suffered this time around from "throwing things over the wall" between teams. In the imperfect real world, design, UI, and server specifications have to

change while implementation is happening (I should say while the developers are learning), and cross-functional teams ought to make this process much more responsive."[93]

The waterfall approach was claimed to be confounded by the belated appearance of UI requirements:

"We should have discovered this months before release, but due to the single threaded nature of our integration tests, and the total absence of load testing except in the last few weeks before the scheduled release we did not. This situation was made 100 times worse by nearly all the of the UI driven data feeds appearing in the last 8 weeks of work forcing us to make a mad scramble to implement, or damage the brea[d]th of features in the release."[94]

3. Lack of server-side control over requirements:

"From my point of view, the biggest problems the Nakamura subproject [had] delivering Q1 had to do with unrealistic assumptions about what could be "thrown over the wall." […] the development process encouraged the UX team to try to figure out wireframes which might cover everything that was needed, and the client-side team to try to figure out a specification which might do everything that was needed, and the server-side team to try to figure out services which might do everything that was needed. And none of us got a chance to actually check our assumptions against Real Life until too late in the schedule.

The UI feature bloat, resulting from the latter approach:

> "We allowed feature scope to increase without changing resources or timescale. This left us with no place to go but to slash quality.... which we did with great efficiency." [95]

The accounts repeated the reasons commonly cited in software development as culprits behind the ailments of software systems. They all agreed that the challenges were a result of poor process, which resulted in poor knowledge. Basically, they brought two types of answers to the epistemic challenge behind the question "Why didn't we find this out sooner". On the one hand, there was knowledge to be had about the system from the technical perspective of performance, but this knowledge was not pursued. On the other hand, there was knowledge to be had about the system from a functional perspective, which had to come from others, but it came late. The accounts associated abstract, structural understanding with software, while functional understandings were associated with concrete features. They acknowledged anticipatory, constructive ways of knowing, but associated them with an external source (the UX team), and with a focus on concrete features. Software-related knowing on the other hand was described to be of a following character, which could work once the system was implemented. Performance results could be used to probe the viability of the architectural model, and to arrive at a conclusion about the inappropriateness of the conceptual model underpinning the software architecture in abstract terms, suggesting for example that JCR was designed for the usage patterns characteristic of the institutional context of Enterprise Knowledge Management, and unsuitable for the usage patterns in higher education. These accounts did not envision an avenue for anticipatory ways of knowing in abstract terms, ways in which the same type of abstract conclusions could have been arrived at ahead of time,

either by comparing the expected usage patterns in the enterprise context with those of higher education, or by mapping the usage patterns in higher education onto the architectural model. Table 7.1 summarizes the epistemic account of server team participants along the dimensions of abstract and concrete knowledge, and anticipatory and following ways of knowing, and suggests a gap in the account in terms of abstract anticipation about the operation of server-side software.

Table 7.1: Overview of the server team's account of epistemic reasons behind Sakai OAE's challenges

	Anticipatory	**Following**
Concrete	*UX design* in terms of concrete UX features	performance testing, QA and pilot
Abstract	GAP	evaluation of architectural model in light of software performance

7.2.5. Making components work together: the kernel team's approach to development

Participants of the kernel team also engaged with server-side software in an anticipatory manner, for the purpose of design, even if these activities did not figure in their accounts. My concern here will be the overall outline of their approach in constructing the architectural model of server-side technologies. A central aspect of this approach was engaging with actual software with the goal of getting things to work. Understanding emerged as coupled with this practical engagement. Assessment, planning and insight

followed in the wake of hands-on efforts to get software up and running. Ian Boston started by exploring the avenues of content management by implementing JCR to work with Sakai 2's kernel. The first implementation in 2006 was tentative, and as such, it was able to behave as both promise and validation, contributing to the formulation of the vision around the new Sakai. In the following, I will describe two episodes with the same outline, one which took place in connection with the server rewrite decision, and another which followed in the wake of the S/OAE performance turmoil.

First episode: not choosing OSGi (and Spring)

K2 got started upon Ian Boston's suggestion of a grounds-up rewrite. The suggestion was shared and discussed in the first weeks of the new sakai-kernel e-mail list, which had been created as a private list for a small group of Sakai participants for the purpose of discussing directions that Sakai 2's Kernel was taking.

> "A thread to scare everyone :) / Here is what I think, we do a complete rewrite but in a way that reuses appropriate code from the past. / Start with an empty container. / For each area of functionality evaluate 3rd party, existing Sakai code and new code and then do integration, improve coverage and code quality or almost as a last resort, write new code. / So although this might feel like a total re-write its more likely to be replacing with 3rd party and re-factoring existing old code. / Does that scare everyone?"[96]

The first announcement was further detailed in a subsequent email:

> "The aim is that we look at what we are trying to achieve, and replace the many of the modules with single off the shelf modules that deliver the concepts.

> For instance: If we believe that a large part of Sakai is essentially a[n] X with customizations for education and research, and we can find a component that delivers this functionality and passes our criteria, then we spend time integrating that component rather than writing our own.
>
> One X that springs to mind is a Content Management System..... the 3rd party component might be Sling.
>
> Another is a Component Framework for which I start to think of OSGi […]" [97]

In the emails, Ian Boston outlined a strategy based on open-source, which was to rely on existing open-source software modules (libraries, frameworks, components and the like) to replace Sakai 2's homebrew solutions. The process involved knowledge about the open-source offerings, and the identification of modules matching the purposes of Sakai. External offerings received priority over existing Sakai code or writing new code. (Note that at this point, Sakai 3 was only in the making, and the email talked about the Sakai platform in general). In case appropriate open-source software already existed, the suggestion was to go with that rather than Sakai-specific code.

Other participants were ready to follow the suggested approach. James Renfro in particular emphasized the advantage of a modular approach with open-source components for engaging with code quality before deployment:

> "I think the primacy of production success as a measure of code quality should be somewhat diminished by appropriate use of unit and integration testing. If we can prove that the code is largely able to do what it aims to do even before we begin QA, then we have a big advantage over the current model. Not to say that we won't have bugs, or that the test cases won't need to [be] revised through a QA

process, but just that we need to get away from this mindset that once one of the big schools has run something in production is makes it mostly "safe" for use by others. We have plenty of critical bugs in code that is running in production at many many schools. ”[98]

Renfro’s comment suggested that the modular approach was useful, because it would allow for a prototype-based testing of integration. This was contrasted with Sakai’s current approach, which meant that new components were locally written and deployed as contrib projects, and they became endorsed by the rest of the community if their quality was acceptable “in production”. He further implied that unit testing and integration represented a superior approach toward quality, which could “prove that the code is largely able to do what it aims to do”, unlike QA and piloting, which focused on bugs, and required that the software be used.

The above framing of the task subtly turned the focus of discussions about the usefulness of open-source components to code quality. Participants had a strong belief in the superior quality of production-tested open-source software over code created within the Sakai community. In the quality of executive director, Michael Korcuska weighed in on this note:

> “Assuming there isn't a good candidate in the "open but custom" category then I'm not worried about the relative lack of JCR implementations. Because the only other choice is to do it ourselves. / And I don't believe [we] will do it better in the time allotted or in the long run.” [99]

Korcuska suggested here that open source was superior to a local solution even if it was the single implementation of a standard, and no alternative implementation were

available for comparison in terms of implementation data. Also, decisions were in large part motivated by negative experiences with solutions available in Sakai:

> "For the relational index on the content store, and other things in the Kernel we need a solid way of storing data in a SQL database. / We know that SQLService has lots of problems, so we will *not* be using that (but it might have to remain to support 2.5.x) / We know that Hibernate has been a nightmare in 2.x Sakai, and everyone who has used it lots has had to be very very careful to make it work."[100]

As this comment by Ian Boston outlined, problems could be of two types. The technology had been known to perform poorly within Sakai (the case of the SQL database), or the integration proved extremely challenging.

Following the suggested approach, the community named a series of component technologies that were considered promising. In the first round of discussions, accounts of first-hand experiences with one content management framework outside the context of Sakai (SpringDM) were pitted against the partial implementation of another framework inside Sakai (JCR and OSGi within Sling). After discussions, skeptics of the second approach agreed to put off decision-making until a proof of concept implementation of OSGi was ready. OSGi was used to load code modules called components, and it was particularly important for getting existing Sakai 2 components to work with the new system, using a single shared kernel for both the old and new Sakai platforms. At the same time, the results were coming slow. One of the skeptics joined the implementation process, but after several weeks of efforts, things were still not shaping up. At this point, participants in leadership positions became impatient, as the expectation was to get the

new kernel ready for the parallel UXI project, which had already started to produce new user interface prototypes.

Many things were made to work in the proof of concept implementation, but the two developers came to the conclusion that integration with the Sakai 2 kernel was too difficult after all, and the tough learning curve they had been going through would mean a significant challenge for Sakai programmers who would need to implement the framework locally. As Carl Hall made the case:

> "As hard as we've been trying to get OSGi to work in different ways, I get the feeling that if it is this much work for us, how much complexity is it going to add for new dev environments. […] I've got some base [basic] things setup in OSGi but have had to really work to get more than HelloWorld going. I can't imagine what fun it will be to go through programmers cafe and teach OSGi. OSGi does present some nice things in classloaders and such but with the effort so far, I'm not sure that we haven't hit a point of diminishing returns. / I know it's been said that we don't want to manage our own component manager." [101]

It was suggested that the component manager was also too heavy for Sakai, involving additional complexity for doing more than what was needed.

The community engaged again in discussion, this time under a slight pressure of bringing the case of the kernel to a resolution. Several alternatives were again outlined. These included a local solution, which was to cherry-pick libraries of code that were really necessary for getting Sakai 2 to work:

"1. OSGi, Ian's efforts are stalled to get K1 work as OSGi bundles.

2. Sling, which includes OSGi and JCR, but no legacy installation on OSGI, using instead [OpenSocial] gadgets.

3. same as 2, but with Shindig [instead of OpenSocial]

4. Mixed approach, mixing Sling's OSGi with Shindig"

Eventually, the two middle solutions were discarded on account of offering an inferior form of integration with Sakai 2. It was decided that the proof of concept implementation outlined in 1 (the OSGi POC, or Proof of Concept) should be matched by a local solution, which came to be called APOC, or Agnostic Proof of Concept, for not relying on specific open source modules.

Since the task was originally framed in terms of picking the right open source component, the nature of the decision had to be reformulated. John Norman provided the updated framing in the following manner:

> "As I understand it, there has been a feeling for some time we should not be seeking to maintain our own component manager when there are good solutions available from other sources. I'm not sure we should be investing our scarce resources in continuing to develop a custom Sakai component manager if it is better solved elsewhere by others. Thus the task is to determine whether others have come up with a solution that is better than ours.
>
> So I would reverse the issue. Can you make a case and demonstrate the argument to say Sakai's requirement is either simpler than the problem OSGi sets out to solve, or that Sakai is fundamentally different from the problem OSGi seeks to solve, and therefore OSGi should not be considered."[102]

This new formulation was saying that "better than ours" involved more than simply being better in terms of overall code quality. To be better, a component also had to do exactly what was needed for Sakai: not more, and not something completely different. It was agreed along these lines that the two proof of concept implementations (the POC and the APOC) will be shared by experienced server-side programmers within their community for their judgment and insight.

Engaging with code: the hands-on practices of server-side software development

The actual proof of concept work involved a range of activities. Before the APOC was decided upon, writing longer junks of code was the least important of these. Developers downloaded components from repositories, and attempted to run them as part of already existing installations. First attempts were typically unsuccessful, the system would not even start up. If it did, the log of the startup could still be reporting errors. If no errors on startup, there could be errors from integration with other components while working with the system. Each step involved a process of debugging: understanding the nature of the errors, finding the possible causes, making changes, and running again.

Reports of difficulties and requests for help related to getting the system to run were very common on the sakai-kernel email list. As these emails suggested, a wide range of things could go wrong: the download was broken or corrupt, or the source was not in good health. The configurations for running on a specific machine, in a specific local software environment had to be adjusted. The order of startup for different modules was often mentioned among the issues: OSGi could run within the Apache Tomcat webserver, but the Apache Tomcat webserver could run within OSGi, and the two required different configuration and startup ordering. If all was fine for setting the system

up, testing the functionalities of the code through queries created as new code could still bring problems to the surface.

To understand these problems, developers had to dig deep into documentation parallel to sifting through the actual code in order to evolve an understanding of the architectural model, where everything was in the code, what different packages were doing, and how flows of operations unfolded from the code base. To do this work, developers would typically rely on development environments, which supported or automated the tasks of installation and debugging. Setting up a development environment could itself be a challenge, and making sure that the system worked with it was also error-prone. Some environments were well suited for some types of code, and developers had a preference for working with environments they were familiar with. Note that at this point the user interface was not added on top of the kernel, so development environments also helped in making the operation of the system accessible to human judgment.

All of the above was typically lonely work, which would surface in IRC and email discussions in case of bigger than usual troubles. Face-to-face collaboration was still considered superior to web-based communication, specifically for learning about new modules or getting a collaborative project going, but opportunities to meet in person were scarce.

Rerun of K2: Sparse Map and Solr

After a long struggle by a growing number of participants, the APOC was put together, and it was selected over the previous OSGi POC. In early 2009, at the time when the 3akai project was launched, the K2 team finally started to work on an integration with the UX, as well as taking requests for functionality. The release of a prototype system was

scheduled for mid-summer, in connection with small scale pilots at Georgia Tech and Cambridge.

Overall progress on the new kernel was slower than expected, and early in the summer, the K2 team decided that it would switch strategies for building the framework, and bring back OSGi, this time within the broader Apache Sling framework. The reasons behind the decision were threefold:

(1) A new decision was made that Sakai 2 did not have to run in the same system as Sakai 3.

(2) Members of the team had interacted with the contributors around Apache Sling, and came to the conclusion that the company behind Sling was truly committed to serving the open source community.

(3) Adopting Sling had supplementary advantages: it made developers' work easier in supporting JCR, and it would allow them to eschew related difficulties encountered in the APOC. Also, Sling came with OSGi, and since difficulties with OSGi were related to getting Sakai 2 into OSGi, these became void without the need of full integration.

In light of these considerations, Ian Boston made a prototype installation, and announced the promise of the initial successes to the kernel team. In spite of the delay which would also cause the cancellation of the summer pilots, the change in strategy was accepted in the summer of 2009. The 3akai project was replaced by the Simple Learning Environment (SLE) project, and the SLE was planned to built on a server-side foundation composed of JCR and OSGi within the Apache Sling content management framework.

I have already told how the performance issues in the first release of S/OAE tore down JCR, a key pillar of the kernel's foundations, and the earliest of its three core

components. The response of the server team followed the pattern of hands-on tinkering that has been described in connection with the proof of concept implementations: Ian Boston and Carl Hall came up with a promising new framework design for file storage, implemented as the SparseMap Content Store module. The new design was presented to the community, and accepted after discussions. The following are excerpts from the emails in which Ian Boston made the initial announcement about the explorations with SparseMap to the K2 community:

> "So, about 5 weeks ago, while waiting for Q1 builds and soak tests to complete I started to look at what it would be like to implement User, Groups, Acl [Access Control List] and content with versioning as directly as possible, keeping an [eye on] concurrency and performance. See the blog post at (1) for details, first in Python and now in Java. The code is in (2)(3) [the numbers indicate urls to webpages]. It uses a very simple storage abstraction, a Column Database or a Sparse Map of Maps." [103]
>
> "If you have been on IRC you may have noticed some ping ponging between me and Carl on a few experimental branches. Although this work is by no means complete it's showing some promise so I think it's worth sharing." [104]

For the coming months, the challenge was to make SparseMap to work with the other components. Ian Boston's midway report testifies to the tentative character of the efforts, and the technical nature of the explorations directed at making things work:

> "there is now a version of Nakamura [K2] that has had the Jackrabbit UserManager replaced with an implementation based on the Sparse Content Map (awful name) storage mechanism.

[…]

What is there still to do: (lots)

A. To make this testable I think we need to port enough of the areas from JCR to the new Interfaces and then check that we have a) concurrency and b) performance. Pooled content might be a low hanging fruit.

B. We need to provide a search mechanism that supports enough of our use cases.

[etc.]

And If A shows that this won't work, we need to go back to the drawing board."[105]

A new search mechanism was one of the technical aspects of making SparseMap integrate with the already existing Sakai 3. Earlier, JCR provided both a content model and an efficient search mechanism over that model to access the files, and now both of these server-side functionalities had to be recreated. Also, an array of interactive UI screens had been programmed at this point, together with the JSON-based queries to the server that they were relying on, so server-side search capabilities had to be recreated for these specific data requests. Ian Boston started by listing possible candidate components for search, but the technical nature of the matter prevented a wider discussion of the capabilities of the technologies. As John Norman commented:

"Got intimidated by the linked page. I don't think I want to even try to understand it :("[106]

Thus, it befell Ian Boston to make the selection, and his next step was again implementation, to test whether the selected component would work. The implementation was announced to the community once it was already to some extent proven:

> "For the UI, the big change is that search templates are now written in Solr rather than JCR-XPath."[107]

7.2.6. The failure of Sparse Map and Solr

When the first official release of S/OAE ran into a performance debacle, SparseMap and search were found out to be the major culprits behind the troubles. The S/OAE project turned to an external consultancy (OmniTI) for an analysis of its approaches. The consultancy's report noted, that:

> "Everything in the system is "search". Any list displayed by the UI is generated behind the scenes via a search query."

Given the mash-up approach of S/OAE, content pages were pulled together from a possibly large number of small pieces of content, group widgets contained up-to-date information about diverse group-related activities, communication and sharing involved the retrieval of personal connections and complex group structures, and all of this was further complicated by nuances of roles within the specific contexts. Search was at the heart of S/OAE, but the consultancy found that its existing technological foundations were severely limiting. Thus, for SparseMap, the following recommendation was made:

> "Consider reducing the overall dependency on the current SparseMap implementation by gradually replacing the various components used on the project with counterparts whose design goals are explicitly restated and match the current application requirements."

SparseMap was perceived as an opaque component for which the design goals were not clear, and whose database architecture was overall unsuitable for the task.

The other severe problem was related to a specific area within SparseMap, the treatment of Access Control Lists (ACLs). As I have said previously, in the institutional setting of education, participation in collaborative contexts was tangled with social categories, like roles, and social contexts were diverse and numerous. In this complex social environment, every search involved the checking of permissions, so the ACL component was as central as search itself. OmniTI's description and recommendation were as follows:

> "Issue: Group memberships, document permissions are all stored relationally in an ACL [Access Control List] table. This representation is not very well suited for ACL data. Querying this data requires multiple logical self-joins, mostly performed in the app, for performance reasons. Scaling such pattern in a relational database will be hard past a certain point.
>
> Recommendation: Investigate the use of a graph database for storing ACL representation, such as Neo4j (http://neo4j.org/)."[108]

The architectural problems described by OmniTI's report had caused the slowness of queries to the server. The problems were experienced by users as pages extremely slow to load. NYU's 2012 pilot started with 3000 users, 6000 pieces of content, 300 groups, as inherited from the pilot of the previous year, and these numbers were duplicated over the first months. Reports of poor performance were collected on a confluence page, and included the following:

> "NYU's Sakai OAE Production Instance (ATLAS) has been running on release 1.0 reasonably well; however, when a group has a large number of participants or a large amount of content, the group page could take a minute or longer to load.

For example: Gallatin Graduate Students have over 200 participants and nursing group with over 100 discussions can take a while to load."

"Gallatin Graduate Students group has over 250 participants. The group's participants widget can take up to ten seconds to load."

"Groups with lots of discussion topics (100+) and replies to each topic (10+) also take a minute or so to load."[109]

The nature of the problem can be further illustrated by walking through Sakai OAE's process of putting pages together upon a user request. The following analysis is my own reconstruction of the operations that were required to create the pages on the fly. Figure 7.3 shows the mockup for a dashboard-style home page from Berkeley's pilot. The menu on the left lists links to dynamically created collections of types of content, like messages, document library, group memberships and contacts. For each of these, a number of items currently in the collection is displayed – each represents a search query, which I assume to work by scouring the available content in order to provide up-to-date counts for the relevant items. In the content dashboard, each widget represents a group of queries of different types: events, favorites and quick links are related to the broader community, while tasks and my links are connected to the person. Again so many queries.

Pages focusing on the display of content could be similarly complex. Figures 7.4 and 7.5 are Berkeley pilot mockups for overview pages of library content and groups respectively. To obtain the page, all items listed had to be found by a search, and we may assume that the varied metadata had to be obtained by one or more different searches for

each item. And the number of items could grow potentially very large, as OmniTI warned:

> "While only the current version of a document is indexed, there are no limits on the number of groups in the system, or the number of entities a document can be shared with."[110]

Figure 7.3: Mockup for the user's dashboard-style home page from Berkeley's pilot

Figure 7.4: Mockup for the overview of library content page in the Berkeley pilot

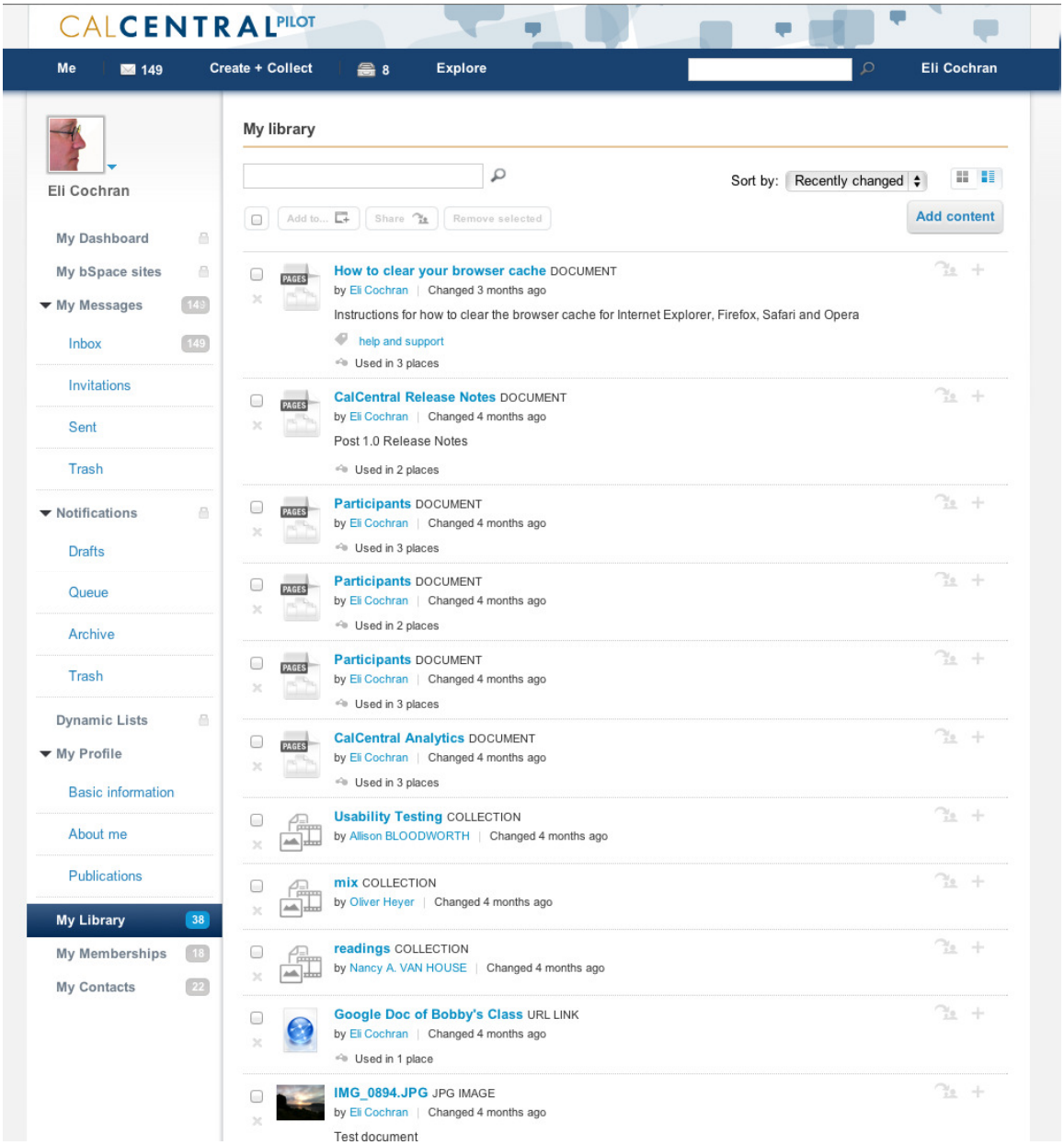
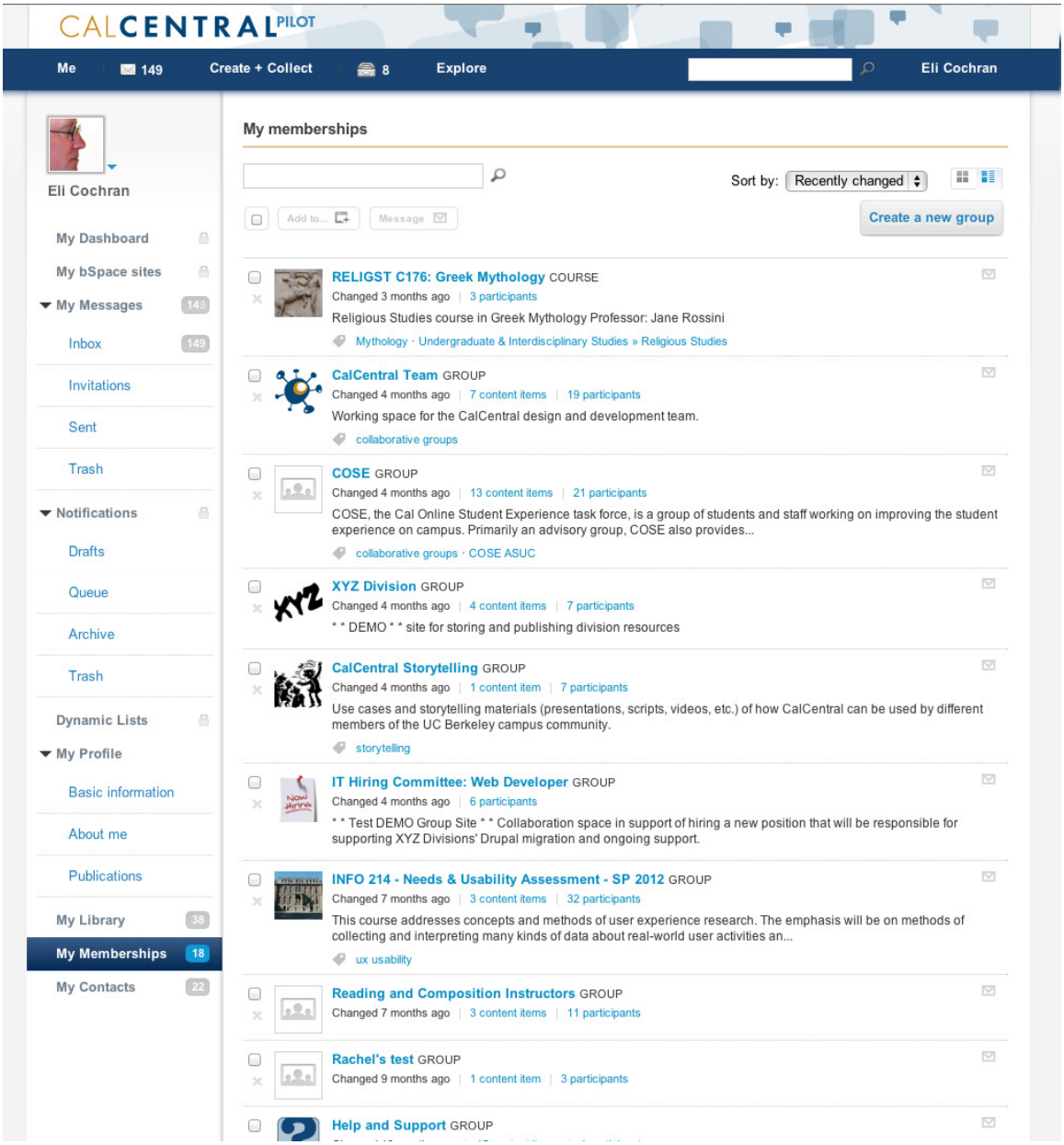


Figure 7.5: Mockup for the group membership page in the Berkeley pilot



This time, the assessment of problems did not involve the types of generalizations that were made by Ian Boston with regard to JCR's suitability for different types of social contexts and their socio-technical usage patterns. Meanwhile, OmniTI connected the patterns of the user interface with the underlying architecture, and pointed out flaws in

the latter in terms of the connection between data structures and search. The analysis found that the information architecture of the kernel was not designed to support the patterns of activities in the user interface.

7.2.7. The disconnect of the back end

The opacity of server-side software

The assessment of both performance debacles found that there was a disconnect between the model of server-side technologies and the patterns of usage in the kind of social and collaborative content platform that its makers wanted S/OAE to become. The problems that guided the initial selection of software components or architectural approaches reflected the educational domain, but they did not provide guidance for the elaboration of the solution. The JSR-170 content management approach was selected on account of problems with Sakai 2 in production, and SparseMap was outlined in light of the social patterns of usage that performance issues brought to the surface. While the connection across technical structures and social patterns of usage could be established retrospectively, related investigations did not appear in the design phases. As I have shown, the design of server-side technologies was primarily technical, relying on prototyping implementations to make various software components work together. In attempting to construct solutions for the perceived needs, the initial strategy was to engage with existing open-source components in a selection process, which relied heavily on prototyping to anticipate the success or failure of the process of integration with existing server-side technologies. When the initial strategy failed, a secondary strategy involved relying on the knowledge obtained from the implementation process to build a local solution.

The excerpts cited previously provide a good sense of the technical nature of the engagement with these technologies. Discussions were often terse and technical, sprinkled with abbreviations or exotic names (OSGi, Gadgets, Shindig) alongside software expressions (integration, component manager, classloader, HelloWorld). An important part of development work was getting technologies to work within an existing context of software. An overall concern was to create clean, well-organized code that would help other developers engage with the software in local implementations. Overall, developers were significantly engaged with the architecture of code, but their conceptual models of structure reflected technical concerns of interoperability and code quality.

These technical concerns often remained opaque to outsiders, and were thus difficult to follow. While developers were working on the first prototypes of the kernel, other members were impatiently waiting at the sidelines, admitting at times that they were "out of their depth" [111] for having an opinion. At moments, sideliners complained of losing sight of the purpose of conversations. Clay Fenlason expressed his struggle to keep up with the work as follows:

> "I'm […] trying to draw together my understanding of where things stand. This is my current impression […] which I'll readily admit amounts to the perspective of an outsider at this point."[112]

Similarly, Michael Korcuska stated:

> "I've lost sight of the high-level goals for a component manager solution at this point." [113]

Clay Fenlason also voiced concerns about his understandings of the direction of the work:

> "P.S. This appears to assume that 3.0 is built on K2, and I'm aware that's still an open question." [114]

His comment suggested that interdependencies across modules were difficult to follow without the necessary software expertise, and it was easy to lose sight of why components were part of a configuration; in the particular case, it was not clear whether and why a new kernel had to be involved in the overall architecture of Sakai 3.

The separation of front-end from back-end

Once the initial selection of components was made, the UX-driven approach resulted in a way of working where the UX team shared JSON data queries that had been incorporated into the user interface. These were called the requirements of the UX, or requirements for short, and became the primary channel through which the server side attempted to align with the educational domain. This way of work had been questioned after the first performance challenges, but the essential practice of passing on data queries from UX to kernel development remained. This was the process even after the introduction of smaller, integrated groups and shorter, task-oriented development cycles. In light of this continuity it may be suggested that the initial separation of back-end from front-end created these practices, and contributed greatly to the disconnect between technical and domain-driven design.

On social metaphors in software architecture

Meanwhile, some server-side developers further contributed to this configuration by their own expectations about the nature of domain-driven design. Server-side technologies came with social concepts embedded in them. JCR's architecture was for example based on the concept of permissions and groups. These software engineering concepts were

well-defined and static. It appears that some developers expected the nature of understandings about the social to be well-defined and static in the same way, rather than evolving conceptual tools to think with. The nature of these differences is highlighted in the following exchange between Ian Boston (lead architect), Ray Davis (member of the kernel as well as the UX teams) and John Norman (manager, with a UX-driven outlook on server-side technologies).

The discussion was started by an e-mail from Ian Boston, who requested a clear documentation of the notion of space in Sakai 3:

> "Hi, / Could someone explain to me what a "Space" is and what the motivation is behind it. I can't remember it being introduced or discussed on list and I would like to be able to make a clear separation between it and a "Group", a "Site", and a Location in the Hierarchy.
>
> It appears to address a specific need, is that need written down anywhere ?" [115]

Space was one of the central terms discussed in the design efforts in connection with making sense of groups and sites within Sakai 3. Responses were accordingly engaging in a cheerful historical account of the evolution of thinking about these notions. John Norman started the line of explanations with an account of the conceptual evolution I have described earlier:

> "'Space' is not a specific need, it is rather a portmanteau term to avoid (extend?) confusion over site/group.
>
> My understanding is:
>
> We said we wanted to separate 'sites' from 'groups.'

Having done so, the designers started adding functionality to 'groups' (what would group members want to do?) to the point where 'groups' were indistinguishable from 'sites' in terms of functionality.

When we revisited the need to separate groups from sites, we (some of us) decided to use the idea of lists (people lists) for the use cases where there was no implied group/site functionality needed beyond a list of names.

So then there was the set of functionality that was once called 'sites', and later 'groups' as the 'groups' functionality grew. Rather than choosing to use either name, for better or worse we chose 'spaces'. So spaces have memberships, roles, pages, dashboards, widgets, etc.

Having done this there was a suggestion that we could reuse the term 'site' to indicate a simple web site/set of pages, that might be content managed by the members of a 'space'. Using this concept would argue against going back to the original use of 'site' for the broader functionality set, but I am not sure this separation has widespread understanding/use yet.

So if 'groups' means to you the functionality set that we ended up with in discussion with Sam about 'groups' functionality and 'sites' means the set of functionality we discussed in the early design work, then space = site = group but NOT = list.

If you adopt this understanding of 'space' (I think saying 'groupspace' might help) then the term 'site' is available for reuse as above, but note that in some current screens, there are 'create site' buttons that create the 'space' or 'groupspace' concept." [116]

John Norman's account mapped out the concepts of space, site and list in connection to each other, and described how Sakai participants constructed their present understandings of these terms by gradually formulating distinctions across them in a conceptual process which was driven by the construction of functionally distinct types of user interfaces. In this description, the meaning of terms was both indexed to historical moments in time (i.e. what was discussed in early design work) and to specific people (i.e. if what group means to you is what it means to Sam). Accordingly, meaning was not only changing and variable, it could also be tentative, and currently in the making ("I am not sure this separation has widespread understanding/use yet").

Ian Boston was not happy with this account.

> "Maybe I am being pedantic, but we already have about 5 meanings for the word Group, and several aliases, and it's quite possible that it won't make the blindest bit of difference, I did try and define Group[2] long before we ended up with 5 Groups, and now everyone just says "Ian's definition of Group", because no one agreed with it. As John said a UI concept of Group got all sorts of things added to it and then that had to be abandoned and renamed "List" which is actually very close to "Group"[2] (Confused, I am:))" [117]

This response expressed discomfort with the fluidity of meaning in terms of evolution, personal perspectives and connections across different concepts. John Norman and Ray Davis each responded by engaging Ian Boston in the type of conceptual debate that they had been pursuing at times in attempting to make sense of concepts. But Ian Boston was not willing to play along. He demanded clarity, fixity and documentation. What is more,

he also demanded that meanings respect existing technical definitions. Thus, his proposed definition of group was relying on software engineering jargon:

> "Support For Groups of users, with metadata associated with each group.
>
> Membership of a group to be either groups or users.
>
> Groups to be part of AuthZ schemes.
>
> Roles to be a special type of group."

This account described groups in terms of the membership metadata associated with them, and defined authentication schemes over the concept. Going further, it suggested that roles were also implemented as groups. In another instance, he was challenging the use of such terms as 'List' on the basis of the technical software engineering definition that he was familiar with:

> "My description of Group [is] certainly too technical and goes into too much detail but I still disagree with naming what is a Group a "List"."[118]

The exchange resulted in a series of thoughtful contributions from John Norman and Ray Davis on the way concepts like 'Group' were being used differently in functional and software-oriented contexts. John Norman described his own contribution as "dancing around a long-standing communication problem:"

> "I think this may be a slightly different issue. For me there is at least one complication with "Group" and that is as far as the back end is concerned a list of people is a list of people and it doesn't matter if that list of people is the membership list of a group (aka Group) or some other list of people. So it is convenient to use the group functionality of Sling to manage lists of people. Meanwhile the UI might start to use Group to only mean the membership list of a

GroupSpace and to use List for other sets of people. It will be a recurring problem that in one case Group = SlingGroup and in another case List = SlingGroup. The backend dev will be constantly tempted to speak about 'group functionality' for both and the UI developer may want different behaviours for the two cases." [119]

This train of thought was picked up by Ray Davis:

"What Ian described very well in the Nakamura document he referred to were "Jackrabbit Group objects as used in JCR authorization checks and as built upon by Sling." (I usually call them "Jackrabbit groups" or "Jackrabbit authz groups" rather than "Ian's groups".) These are part of implementing "human-group"-related requirements, but by no means exhaust the subject."[120]

In a subsequent document, Ray Davis attempted to provide further explanation for this issue:

"Jackrabbit is a content repository and Sling is a web framework. They aren't social networking application suites. (That's our job.) When we talk about a Jackrabbit or Sling "user", we don't mean a person with a name and an age and an address: we mean that we can get some opaque identifier from some sort of authentication, and we can use that thing to assign and check access rights (and as a container for properties of our own). A Jackrabbit "group" is only an identifiable pointer to those kinds of "users" and to other "groups", and its only inherent use is also as a way to assign and check access rights (although again we can attach properties of our own). Jackrabbit "access rights" don't include "revise the standard syllabus" or "read final grades" or "broadcast a message to the teaching assistants" or "moderate a chat room". They're a static set of file-system-like

> actions that can be taken on file-system-like nodes: read, update, delete, add children, remove children, read the access rights, and change the access rights. Obviously there's a big distance from this bare authentication-and-access-rights skeleton to real-world people and communities. But I don't want to assume that the only way to model the richer domain is by warping that perfectly useful skeleton."[121]

These accounts hinted at two important problems in connection with applying the social concepts of software development for modeling software. First of all, the use of these concepts within the software development community had become fixed in a way that was indexed to technical operations. The design of the user interface allowed for negotiations around the use of these concepts, but software development did not. Second, the software engineering concepts were in correspondence with relevant phenomena of the social world, but they were "distant" and "by no means exhaust[ed] the subject."

Overall, the accounts suggested that software development may be prone to rely on a series of received concepts for making sense of its social context, and what is more, it may hold the expectation that the social context will be aligned with these received concepts since they have become incorporated in software technologies. However, as Ray Davis concluded, software design should not start from existing models of software to see how they may be used in the real world, as this would only result in warping (and breaking) the models. The conceptual models of software architecture may provide connections to the social world, but overall, the models underlying software implementations could not be used directly to frame and support the conceptual construction of social aspects of software.

Subsequent emails from Ian Boston with regards to UX-requirements suggest that the commentators were not mistaken in their account of the expectations of software developers for the social context to be aligned with their conceptual models. When the UX-team presented an overload of requirements to the kernel team, Ian Boston's reaction was as follows:

> "how do we decide what is possible and what needs re-work to make it possible? […] Factors that might influence what is possible are: […]
>
> Technology already adopted.
>
> Available technology.
>
> Technology we might be able to invent."[122]
>
> "At the moment there is almost no contact in the UX design process […]. Consequently the 2 specs I have seen (Spaces and Dynamic User Profile) are disconnected with what comes naturally at the back end, which just makes them cost more and take longer to implement."[123]

In these complaints, Ian Boston raised the legitimate concern that the designs of the UX team did not reflect the server-side, and threatened to break the software because of this. His suggested solution was that the user interface should be designed with server-side technologies in mind, reflecting "what comes naturally at the back-end". In line of his earlier discussions related to the notion of groups, this was not a matter of decision-making power, but was instead based on the belief that software technologies represented consensually agreed upon notions of social concepts, such as "Groups." Once these technologies had been accepted, the underlying conceptual models had been taken on,

and this represented a binding agreement which the UX-team should have respected in their design work.

Overall, the exchanges related to conceptual models and requirements suggest that a major source of disconnect between technical and domain-driven approaches was in their understanding of the engagement with conceptual models of the social context in the process of software design. From the domain-driven perspective of the UX, concepts evolved in a collaborative sense-making process that was rooted in the "use cases" of education. From the perspective of server-side development, conceptual models of the social context had been defined and built into technologies, which meant that this area neither required, nor allowed for further changes, as these could result in suboptimal use or failure of the available technologies.

7.3. Discussion

As I have shown, S/OAE was evolving into a system based on dynamically produced presentations of smaller units of content. This approach required efficient server-side storage and retrieval solutions for units of content *and* meta-data. The project used a mix of open-source and locally built back-end technologies to support these content-management ambitions. However, the design of the back-end did not engage directly with the contexts of use; user activities were instead only envisioned in the process of designing the user experience, and came to impact the back-end in an indirect manner. The range of actual search queries that the back end was expected to satisfy was shaped by the user interface, which itself represented novel forms of engagement with content. Some of the novelty of this engagement was however not anticipated by the server-side

team, which resulted in poor performance, specifically in connection with how broader patterns of usage impacted the retrieval of content by means of search. The diagnosis was a poor design of information architecture.

In the various accounts reflecting on performance challenges in the two releases of S/OAE, five conceptual threads appeared in the way the architecture of the kernel and patterns of usage were connected.

1. the nature of individual searches;
2. their distribution and co-occurrence on a dynamically updated, mashup or dashboard style interface;
3. the patterns of interaction with different types of content over time;
4. the volume and topology of the emerging pool of content;
5. interconnections in patterns of activities across varied social contexts of use.

The first two threads focus on the user interface, and the kinds of user interactions that it makes possible. The remaining three layer temporal and social dimensions over this basic view of user interaction, which result in social and temporal patterns of use by various groups of people over sustained periods of time. Thus, threads 1 to 4 were implicated in conceptualizing a scenario where course deadlines resulted in peaks of interaction in areas of the UI like discussion, and the temporary rush of queries from a large number of users (3) was exacerbated by the implication of a high number (2) of complex search queries (1) over a growing volume of discussion content (4). What's more, two separate courses could follow the same deadline on the same type of task, without mutual access to the relevant content, and moderators could be given permissions to moderate for a subgroup of the course section (5).

In Chapter 6 I have shown that the user-centered approach of the new Sakai involved a perspective of the system of user interactions consisting of group-related interactions with a user interface. The above conceptual threads outline a different type of system, with a social and temporal character. Conceptualizing this system would have required seeing the system of user interactions as part of a broader systemic view, which incorporates a range of recurring interactions in connection with content and content management software, and exposes patterns of interactions characteristic of the educational context in connection with the architecture of the underlying technology. This broader systemic perspective went unrepresented across the UX and kernel teams in designing the platform. It eventually emerged around the time when the platforms were already being used, or were close to becoming used. The fact that the problems were eventually interpreted in terms of broader systemic connections between patterns of use and software architecture suggests that conceptual connections between the educational context and server-side software technologies were indeed possible. However, the perspective only emerged after the software platform had materialized in an advanced form; during the phases of design, there was no attempt to invoke the imagination to foresee how the architectural model of the platform would play out with respect to the patterning of activities in the educational contexts of use.

In the following I will argue that the conceptualization of these patterns has been the epistemic terrain of sociology, where it has been described as a way of thinking characteristic of the field. It has been referred to as the sociological imagination, sociological perspective or sociological thinking. Meanwhile, the systemic view required by design demands a specification of the broad epistemic stance of sociology with respect

to technology, in this case software. I will suggest to use the term 'social-technical imagination' to describe this design-oriented perspective, and 'social-technical gap' to refer to the absence of the perspective in design. I have deliberately chosen 'social-technical' to mark the distinction from the term 'socio-technical', which has been used to describe a particular approach to organizational design. Social-technical is meant to denote any kind of conceptualizations of patterning in human activity in connection with technology, which is in our case software. The term also underlines a dialogue with human-centered computing, where it has been used by Ackerman (2000) to describe the inevitable gap between two orders of realities, the social world and the technical domain of software.

The notion of a sociological imagination has been used to mark the terrain of sociology amidst (and despite) ongoing theoretical debates about the ontological status of social phenomena. In Thinking Sociologically, Bauman and May (2001) describe the epistemic stance of sociology as follows:

> "The concerns of economics and political science are by no means alien to sociology. […] Yet sociology, like other branches of social study, has its own cognitive perspectives that inform sets of questions for interrogating human actions, as well as its own principles of interpretation. From this point of view, we can say that sociology is distinguished through viewing human actions as elements of wider figurations: that is, of a non-random assembly of actors locked together in a web of mutual dependency (dependency being a state in which the probability that the action will be undertaken and the chance of its success change in relation to what other actors are, do or may do). […] so figurations, webs of

> mutual dependence, reciprocal conditioning of action and expansion or confinement of actors' freedom are among the most prominent preoccupations of sociology." (p. 5.)

May and Bauman suggest that sociological thinking is characterized by a focus on conceptualizations of "figurations of human actions" that unfold in "webs of mutual dependence" and "reciprocal conditioning." Besides an emphasis on figurations arising across individual actions, the characterization points out the connection between the individual and these broader patterns.

The theme of connecting the individual and the social also appears in the account Mills (2000/1959) gave of the sociological imagination in his programmatic book bearing the same title:

> "The sociological imagination enables us to grasp history and biography and the relations between the two within society. That is its task and its promise." (p. 6.)

In these accounts, sociology emerges as a way of thinking that connects individual action with the social perspective, and seeks to build conceptual models of social patternings or figurations in activities in connection with the individual's perspective. Thus, Giddens' Introduction to sociology (2009) presents sociological thinking as unfolding from an individual's activity of drinking coffee to incorporate the symbolic underpinnings of a social ritual, the normative implications of social regulations of drugs, the global economic relations of trade, a cultural history of diets, and political debates about globalization and the environment. As the latter depiction suggests, sociological thinking is diverse and fractioned into theories of the social and domains of interest (Ritzer, 1975); my point here is not to argue for the unity of sociology in terms of an underlying pattern

of thought, but to outline an epistemic stance that could be useful for conceptualizing the social-technical character of software systems like the new Sakai.

Despite the common use of the term 'imagination' to describe the epistemic approach of sociology as a science, the concept of sociological theorizing cannot provide a model on its own for the context of software design. The conceptual models of sociological theories are not directed at a future artifact. Because of this, they do not partake of the possibilizing mode of thought underlying design, and they do not require a social-technical systemic view which implicates technology in social patterning. While the sociological imagination in and by itself is not sufficient to guide design, my claim is that the epistemic stance of sociology can become the foundation of a social-technical imagination, which seeks to make sense of possible future forms of software as a system that brings the architecture of code in connection with patternings of human activities. To seek an understanding of what social-technical imagination may involve beyond an interest in the patterning of human activity, I will turn to the cognitive-epistemic analysis of Sakai 3's software design.

In the previous chapters I have shown that the design of Sakai 3 involved a distributed cognitive process of conceptual modeling. Participants were focusing on building a new generic approach for the user interface of Sakai 3 which would give centrality to groups in the overall user experience. Conceptual models revolved around typical scenarios of individual users' interactions with possible forms this new interface may take. They were reflective of the broader context of education, and involved an understanding of the nature of its social activities, such as collaboration or networking. At the same time, it was only the meaning of the activities that was derived from the

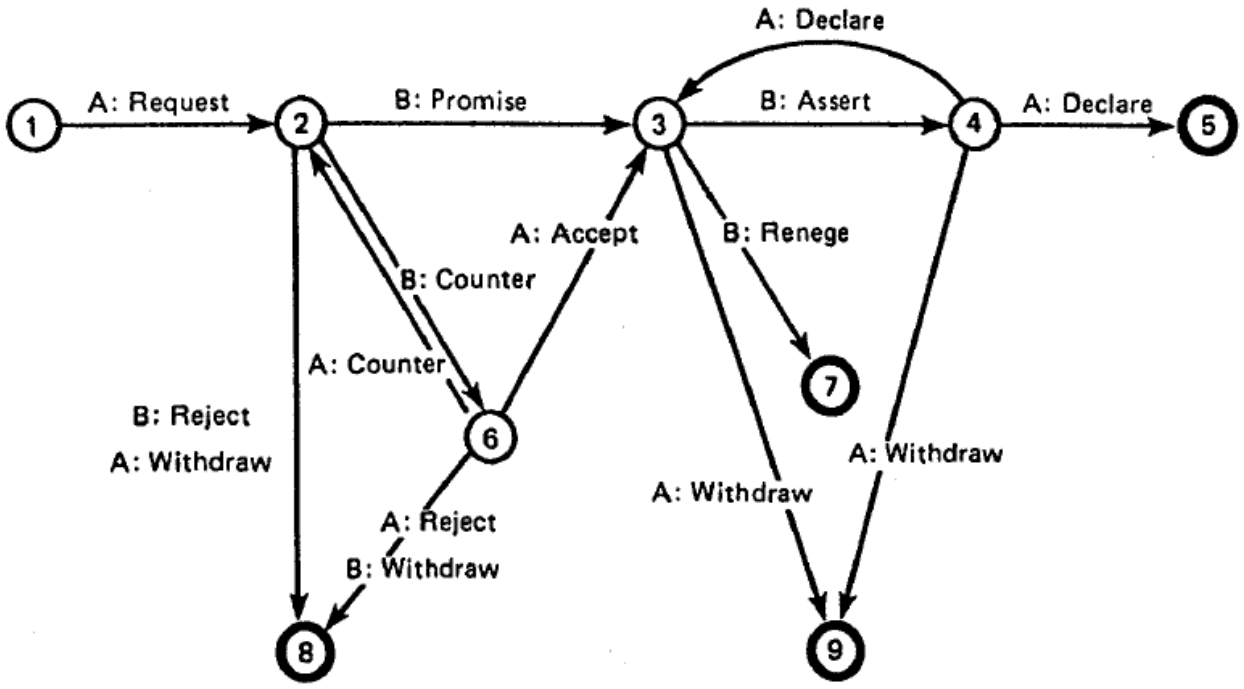
broader social context, the activities themselves were envisioned as the individual user's interaction with the user interface. The imaginings were often rooted in prototypes and other experiential manifestations of the user interface, but they could also become highly abstract.

A social-technical approach may be generally conceived as an analogous process of distributed conceptual modeling, with a focus on the social-technical system. The analysis of Sakai 3's design further suggests that prototypes could play a central role within this process, connecting the construction of the material form with anticipatory modes of thinking about its use. In case of the user-centered approach, the system of user interactions was outlined in connection with these experiential manifestations, which also served as rudimentary proofs of concept for the feasibility of their more complex programmed counterparts. It appears that the social-technical system of a software platform like Sakai lacks similar prototypes that could serve as framing devices for conceptual modeling. I will now turn to what a framing device for the social-technical perspective may be like. I will start by surveying various conceptual tools that have been implicated in the design of software from a social perspective, and point out their limitations.

The first case in point is the use of scripts and categorizations by Winograd and Flores (1987) to amplify existing forms of patterning in communication between individuals, and Suchman's widely cited critique (1993) of the attempt. Winograd and Flores used a combination of scripts with speech act theory for their collaborative software called the COORDINATOR. As Figure 7.6 reproduced from their work shows, the coordinator used a pre-established script of equally pre-established speech act

categories to provide meta-level guidance to communication in the office. Their stated goal was to provide guidance to desirable forms of communication which had been established on the basis of social research. Searle's theory about the existence of categories of speech acts (1976) and Schank and Abelson's theory of scripts (1977) both enjoyed significant scholarly attention in their time. Both of these theories posited that we understand what other people are saying because we rely on pre-established patterns of action, such as speech act sequences like asking and answering, or requesting and responding in Searle's theory, or typical sequences of activities like the restaurant script of entering, ordering, eating and exiting in Schank and Abelson's account. As Suchman suggested in her critique of the COMMUNICATOR, as well as in her related theoretical work on Plans and situated actions, these theories became distorted in their application to software, insofar as the categorizations they pointed out in human activity came to be seen as generative of meaning in a manner analogous to the operation of computers, and they were also applied to software design along these lines. However, as Suchman pointed out, such cognitive constructs (among which she also counted plans) should be understood as resources implicated in our sense-making, rather than sources from which meaning could be automatically generated. Suchman's critical analysis does not apply to the actual theories, but their translation to the world of social software. From our perspective, she shows that sociological theories about the patterning of human action cannot be used as framing devices by means of a direct translation to the algorithmic foundations that govern software.

Figure 7.6: The basic flow of conversational activity in Winograd and Flores' COORDINATOR software (cited in Suchman 1993)



Another case in point is the recent interest in standardization for interoperability in health-care software platforms, exemplified by such large software development communities as Health Level Seven (HL7), NHIN Direct and NHIN Connect. The goal of these efforts is to create standardized protocols of data exchange and implementations that enable the exchange of patient data across different points of service and providers in health care. These systems wrap existing nomenclatures and categorizations in health care in broader protocols of communication, which arrange data in pre-established patterns, and also handle the process of exchange across computer systems. These efforts have been heralded as the promise of major improvement in the quality of health care, notably on account of the standardization of practices (Committee on Quality of Health Care in America, 2001). Meanwhile, ethnographic studies of standards in use suggest that

the standardization of local interpretations and practices does not follow from the use of standards (Timmermans & Berg, 1997, 2003). It appears that the problem of shared information spaces in health care may be better approached in terms of boundary objects and information infrastructures, which have been described to connect heterogeneous local practices and interpretations. Timmermans and Berg have further argued that medical data is entangled in contextual interpretations of its production, and further work is required to make it transportable across contexts. These accounts suggest that the social design of information spaces should take into account local usage, while interoperable software is being designed with the opposite goal of aligning local contexts with an information space. Interoperability standards may work well to guide the design of compatible software systems, but they are not good candidates for a social-technical framing device, as they make no attempt to provide handles for the exploration of user engagement with information in the local contexts.

The modular approach emerging in the open source arena brought about a common separation of back-end from front-end. This strategy was also emphasized in the design of Sakai 3, and I have argued that it created barriers to the anticipation of social patterns of use and their implications for the resulting data. The architectural model of back-end frameworks may incorporate metaphors about the social world, such as permissions and groups. The case of Sakai indicated that these social metaphors led back-end developers to think that they were engaging with the social context. At the same time, my analysis has suggested that these concepts had acquired a fixed meaning within the software domain, which prevented them from becoming implicated in dynamic, explorative forms of thinking. The concepts of group management in K2 were not

working to spark an anticipation of user activities, not to mention their patterning in connection with the "groupiness" of users in the educational context. On this account, they were not able to serve as hinges that could connect technological tinkering with an anticipation of unfolding patterns of use.

In sum, it appears that social theories, interoperability architectures and social metaphors have not been able to contribute to the creation of framing devices that could connect the technical and social design of software systems. I will now turn to the consideration of characteristics that framing devices for the social-technical imagination should have.

The content management and search components that K2 was relying on belong to a broader category of back-end software on the web, which contribute to a patterning of user activity. The broad domain referred to as social media or social computing has been created on top of such solutions. User experiences in social media significantly rely on data models and algorithmic solutions that define how the input of participants becomes aggregated and fed back to others. As Erickson (2013) suggests, "social computing is not so much about computer systems that accommodate social activity, but rather it is about systems that perform computations on information that is embedded in a social context". In the social web, content participates in two logics: the logic of meaningful user interactions on the front end and the logic of algorithmic operations on the back end. The search capabilities that content management platforms incorporate similarly participate in a double logic, where search algorithms and provisions of user interaction work in tandem to create the user experience. The confluence of the two logics materializes at the level of data, which arises from user activities, and becomes

implicated in algorithmic search operations. My analysis of the failure of K2 suggests that grasping the logic of these platforms requires an appreciation of the various components at the same time. A similar approach is reflected in the strategy of incremental design adopted by many social media platforms. Incremental design means that platforms are being tweaked and redesigned while in use. This approach promotes the actual, live system to the status of framing device. At the same time, incremental design turns social media itself into a social experiment, which implicates users in acting out their own possible futures. What I am after here is a framing device which would work similarly to these systems, but in an anticipatory, prototyping mode, creating the conditions for anticipatory forms of experimentation that do not implicate the users.

An important characteristic of live systems as framing devices is that they present designers with a dynamic outlook on the software system, where tinkering with the system results in dynamic flows of events. In the context of design research, Nersessian has described a similar process, where experimentation results in the construction of dynamic prototypes which become implicated in distributed conceptual modeling. This process has been described as cognitive partnering with experimental devices (Nersessian et al., 2003; Nersessian, 2012). Here, I will describe two prototyping tools from game design which afford similar processes of dynamic exploration, with a focus of conceptual exploration in design rather than theory building.

Salen and Zimmerman (2004) suggest that games are emergent systems, which present designers with the challenge of designing them on two levels, in terms of rules, and in terms of the behaviors that result from these rules:

> "Designing an emergent game system that generates meaningful complexity from a simple set of rules is challenging. Why is it so difficult? As a game designer, you are never directly designing the behavior of your players. Instead, you are only designing the rules of the system. Because games are emergent, it is not always possible to anticipate how the rules will play out. As a game designer, you are tackling a second-order design problem. The goal of successful game design is meaningful play, but play is something that emerges from the functioning of the rules. As a game designer, you can never directly design play. You can only design the rules that give rise to it. Game designers create experience, but only indirectly." (p. 168)

Mathematical analysis has been used to analyze some rule sets to model the possible outcomes, but computer games often combine radically open rule sets with visual interfaces rich in meaning, which represent a broad range of possible emergent outcomes. This is a characteristic they share with the social web, which Geert Lovink (2013) has aptly characterized as "a semi-closed ephemeral space". The two examples of software I will present below allow designers to engage with games in the process of design both as rule-sets and as emergent outcomes, albeit in a radically different manner.

Game-O-Matic is a web-based game generation tool that has been constructed by researchers at GeorgiaTech to support the user-friendly creation of games. Users work with a simple graph-based concept map representation of the game, which allows them to connect objects through basic 'actions'. Objects can be defined by a textual or visual representation, and they can be anything the user wishes. Figure 7.7 shows the example of setting up a game with 'cat', 'dog' and a 'dog catcher'. In the final game shown in

Figure 7.8, cat and dog are represented with graphics, ‘dog catcher’ is just a textual description. Actions, on the other hand, are hard-coded in the system, and can be picked from a list, which include basic interactions like avoid (shown in Figure 7.7), chase, eat, push, catch, etc. The actions have been pre-programmed with a basic visual behavior: ‘chase’ results in the object always following the other, while ‘avoid’ evokes the opposite behavior.

Figure 7.7: Setting up a new game with the help of a graph in Game-O-Matic, as presented by the instructional video for the software

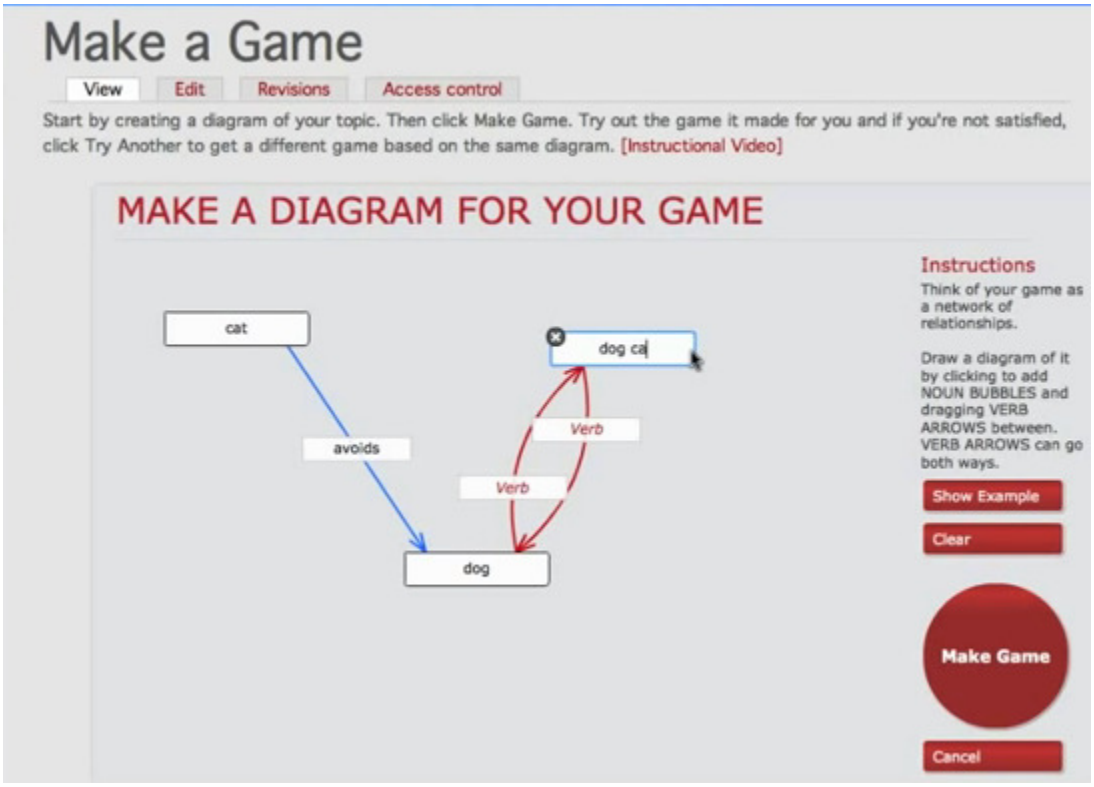
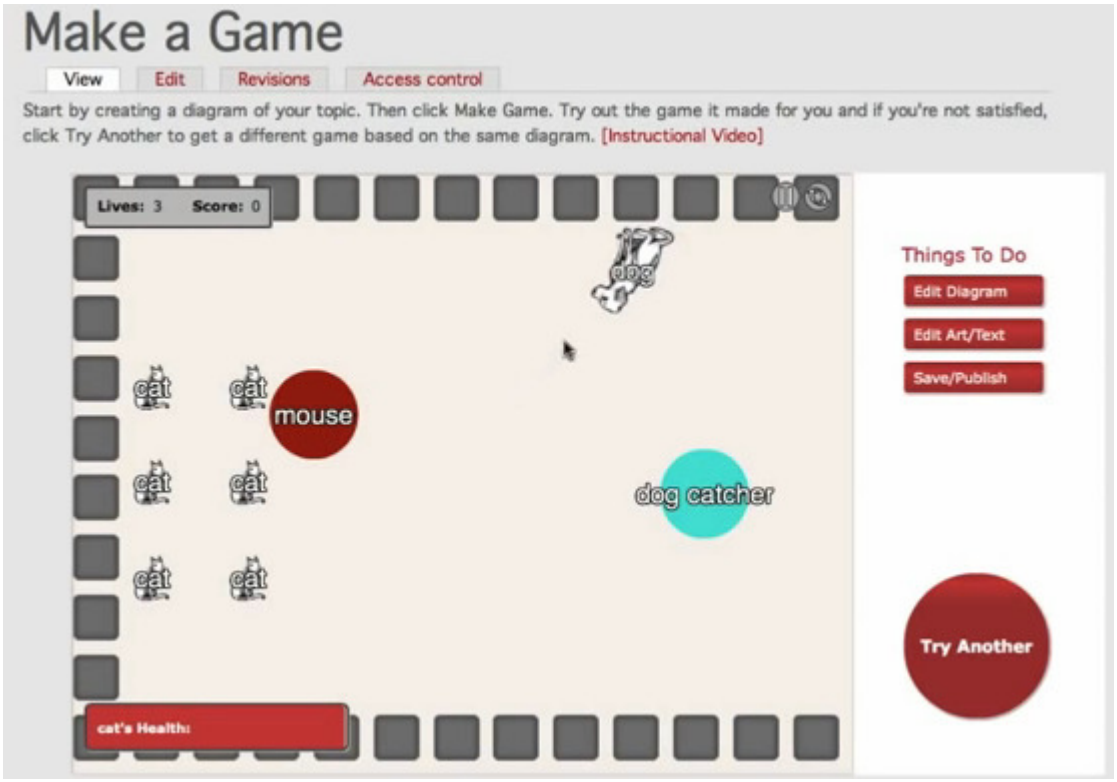


Figure 7.8: The new game in Game-O-Matic, as presented by the instructional video for the software



Game-O-Matic scaffolds game design on the basis of a graph-based model of the game, whose elements are connected to pre-programmed game dynamics. All of this is of course very basic and limited in comparison to the richness and complexity that digital games can achieve. Meanwhile, the tool was designed for creating journalistic games which serve to illustrate the dynamics of conceptual relationships in contemporary news contexts through game play, and it can also be used to prototype game ideas and test their interestingness in game play. The concept map serves as a hinge representation, connecting software operation with the game space. While in this case the tool serves to hide the software mechanics underlying the operation of the game, a possible extension

could be to allow users to engage with the underlying code, and add new programmed behaviors.

Game-O-Matic supports the quick design of games, but provides no guidelines for evaluating gameplay. The second example, taken from Salen and Zimmerman (2004: 166-167), will focus on this aspect of the process. It is a game design environment which has been specifically developed for the purpose of evaluating how tweaking the code of specific game behaviors affects game play. The game in question (Gearhead) is about throwing toys with characteristic programmed behavior onto the opponent's terrain: the fast but erratic cockroach, the kangaroo which punches other toys, the slow-moving giant robot, etc. The testing environment is a playable prototype, where designers could insert new toys, tweak the attributes involved in their behaviors, and start a new game to play test the changes. Besides the actual gaming experience, the environment also provided statistics about the outcomes, which allowed for comparison across various settings. Salen and Zimmerman further describe how the environment helped designers discover that combinations of toys could have interesting effects, a feature which was not part of the original description of the game, and emerged during design. Combinations allowed for a novel approach to playing the game relying on strategies of using various tools for an emergent effect. While some combinations were tested by designers, the use of combinations remained a truly emergent feature of the actual gameplay.

Both game-design tools I have described have been designed to allow for assessing gameplay in light of changes in game dynamics. Both support the setup and tweaking of game dynamics in a way that quickly connects changes to gameplay. Based on the accounts about the use of the two design tools, we may assume that an

understanding of the good game is constructed in repeated cycles of tweaking and assessment. What's more, both design tools provide conceptual representations which scaffold this mental process of construction, and serve as framing devices in this respect. Game-O-Matic focuses on a flexible initial representation of the game as a concept map, while the Gearhead test environment focuses on a representation of the gameplay itself in terms of statistics. I will have to add that these representations achieve their status of framing device on account of their embedding in more complex design tools, where they can serve the thought processes of constructing a complex understanding of the game in terms of the interplay of programmed behaviors, user experiences and emergent game processes.

In an analogous manner, framing devices for social-technical design would need to connect operational models of information architecture with meaningful user interactions and the emergent social playfield. I have suggested that conceptual models drawn from social theory, operational models of software architecture and social metaphors have not been successfully used for this purpose. In light of the game design examples it may be suggested that a conceptual hinge is needed which allows for tinkering with the particulars across programmed behaviors, user experience and emergent field of play, and thus affords modelling across the three domains.

The notion of social-technical imagination should in turn be understood as the thinking process that unfolds around such framing devices. The use of the term imagination highlights the constructive aspect of this conceptual process. It also marks the connection to sociological imagination, which I have described as an epistemic stance that is able to hover between the particulars of individual action and general forms of

sociological patternings. This thought process is social-technical in that it connects the technical domain of software process with the social domain of user meanings and patterns of use.

In the analysis of K2's construction, I have described the lack of imagination bridging the social and the technical in software design as an epistemic gap in anticipatory ways of knowing. I will now propose to call this gap the social-technical gap. This is a response to Ackerman (2000), who described the social-technical gap in terms of the incommensurability between the realities of the social and technical domains. Ackerman suggested that despite the fact that the gap cannot be closed, social research should be attentive to its nature. I have proposed here an interpretation of the social-technical gap in epistemic terms, which suggests that our attention should be focused on the epistemic processes that allow us to bridge the gap in the course of design, and to design socio-technical systems relying on conceptual foundations cutting across the social and technical domains.

CHAPTER 8. KNOWING THE CONTEXTS OF USE – A DISTRIBUTED COGNITIVE-EPISTEMIC STRATEGY

8.1. The empirical nexus: mediating contexts of use to software design

> "Basic research on human organization and action is relevant to office information systems insofar as innovation in design is tied to innovation in the underlying conception of the activity that the design supports." (Suchman, 1983: 327)

This is how Suchman closes her outline of a social approach to the design of software systems in her 1983 paper *Office procedure as practical action: models of work and system design*. The quote suggests that novelty in software design follows from innovative changes of the conceptual models of the relevant human activities. The case studies in this chapter will describe a design process along these lines, showing how innovation in software is a result of conceptual construction from understandings rooted in the contexts of use.

In her paper, Suchman is also making the argument that software design should rely on the study of the judgmental practices that produce the procedural orderliness of work:

> "the recommendation is that only a rigorous program of disinterested study can guide the long-term development of technologies more genuinely supportive of the work that actually goes on in the office" (p. 327).

It appears that human-centered approaches to software design have mostly defined their contribution to design in terms of this latter form, focusing on an empirical nexus with the contexts of use. This approach may be seen to be rooted in the conceptual

configuration of early HCI research, which focuses on fitting the system to an external reality (the world or the context).

User needs has been a central concept for the latter strategy. The accommodation of user needs is a dictum that has been emphasized by a range of different practices, with the result that the concept of user needs has become the general currency in software design, to the point where it has come close to being vacuous. I will limit my survey of the concept to explicit, theoretically oriented accounts, which will not allow me to claim to address all the ways in which the notion has been used. These theoretical framings have greatly influenced the widespread approach of considering users and more broadly the contexts of use as an outside reality for the purpose of designing software.

Within software engineering, the notion of user needs has been straddling the two perspectives of utility and usability (Grudin, 1992; Nielsen, 1993; Löwgren, 1995). These perspectives, as Grudin pointed out, were respectively rooted in the professional communities of in-house and commercial development, and corresponded to the academic communities of Management Information Systems and Human-Computer Interaction. Nielsen (2006) distinguished between utility and usability as follows:

> "Utility is the question of whether the functionality of the system in principle can do what is needed, and usability is the question of how well users can use that functionality." (p. 25)

The user needs underlying utility relate to the functional aspects of a software system, and how they participate in achieving meaningful human goals. This type of user need resides with people in the contexts of use. It has a volitional and projected character, but in case it is expressed as a demand or requirement by a client, it can appear as a

constraint, since new software will have to conform to what is "needed" by the client. User needs underlying usability[h] relate to generic or specific human capabilities, and the limitations that they impose on the way a functionality is achieved by a system. This type of user need has been typically interpreted as a constraining influence on design. At the same time, DCog has argued that cognitive software artifacts augment human capabilities (Norman, 1991; Nardi & Miller, 1991; Hutchins, 1995b; Hollan et al., 2000; Liu et al., 2008). In this manner, human capabilities may also lend a possibilizing character to design.

Both of these perspectives have contributed to an understanding of user needs as real world constraints that the design of software systems need to take into consideration to avoid failure. User needs have become implicated in an empirical nexus to design, as entities or aspects that are out there in the world, and need to be studied or known to inform software design: requirement-type user needs are elicited and gathered from stakeholders, and usability relies on observational studies of people in concrete situations or empirically grounded general principles about human capabilities. The notion that needs are out there in the world, exercising a limiting or constraining power on the range of technically possible designs is further expressed in accounts that talk about the fit of systems to human contexts and needs.

As software was expanding beyond single-user interaction and into more varied contexts of use, the psychological and cognitive edge of HCI way to social approaches

[h] Donald Norman has for example consistently applied the term user needs in connection with usability in his popular book The Psychology (later changed to Design) of Everyday Things (1988, 2002). This usage is also apparent in special needs and accessibility needs.

(Rogers, 1997). Social itself appears as a term with various connotations that I will only briefly mention here. In connection with software, it has been used in the following ways:

1. to refer to the implication of more than one person in the interaction with computers, as in CSCW;
2. to highlight the importance of societal arrangements, like organizations, and social institutions for situating computer use, again in CSCW and Social Informatics;
3. to point out the relevance of socially constructed meanings in connection with the diversification of human contexts, as in ethnomethodologically oriented approaches.

In each of these perspectives, the limited focus on users and user needs gave way to broader framings in terms of social and organizational context, which I will generally refer to here as the contexts of use.

At the same time, social approaches often inherited the epistemic stance of the earlier usability and user needs approaches, including the empirical nexus to the contexts of use and the call to achieve an adequate fit of software systems to pressing social phenomena. Most famously, Ackerman described the socio-technical gap as "the divide between what we know we *must* support socially and what we *can* support technically" (2000: 179), and argued that computer systems will never be able to fully conform to, and even less supplant the flexible, nuanced, contextually situated character of human activity. He also suggested that understanding and dealing with this gap should be placed at the center of CSCW research (see also Dourish, 2006).

Related to the empirical study of human contexts, ethnography has come to occupy a central position within social approaches to the contexts of use, so much so that theoretical points of discontent and disagreement have been commonly discussed in

connection with the ethnographic method (Anderson, 1994; Button & Dourish, 1996; Button, 2000; Dourish 2006). In the following, I will describe the empirical nexus of social approaches to design across the spectrum of different positions on the use of ethnography in software design.

Ethnographic methods based on contextually embedded, situated observation and personal interviews have become common in professional practices that follow the legacy of usability and requirements engineering. Within the methodologies of Contextual Design (Beyer & Holtzblatt, 1998), Interactive Design (Cooper, Reimann, Cronin, 2007) or Goal-Oriented Design (Cooper, 2004) ethnographic methods have been used as empirical tools for collecting needs and requirements from end users and obtaining understandings about the contexts of use. With the help of ethnographic methods, researchers have been able to obtain rich and reliable data about the contexts of use, which could be subsequently edited in various formats for the benefit of design. Sommerville, Rodden, Sawyer, Bentley and Twidale (1993) have argued in a similar vein that ethnographic methods could inform the requirements engineering process by providing insights. The various ethnographic approaches could be used to understand how the social context presented constraints and opportunities for design (a legacy of the usability perspective), or to understand what end users want (a legacy of the utility perspective). Related to the latter, strands within participatory design have been active in giving voice to end-users with the help of ethnographic methods.

These professional practices have been critiqued (Anderson, 1994; Button & Dourish, 1996; Button, 2000; Dourish, 2006) for a lack of analytic and social theoretical orientation, and it has been suggested that they unjustifiably labeled ethnography what

was in fact no more than fieldwork. Button called the approach scenic fieldwork (2000) to indicate the descriptive orientation underlying these methodologies. At the same time, more analytically and theoretically oriented approaches have still shared in upholding the empirical nexus mediating the contexts of use to design (see Figure 8.1 below). I will now turn to the status of empirical research within the more theoretically oriented approaches.

Anderson (1994) has argued for continuing the intellectual tradition of cultural anthropology, which seeks to build conceptual accounts of local logics as they emerge from the meaningful practices within a culture. While fieldwork-type approaches attempt to convey understandings about a setting, cultural anthropology first does the work of analysis on its collected data. Cultural anthropology locates the work of theory in the kind of analysis which is bound to and emerges from local meanings of culture. The role of ethnographic reports is not so much in the immediate usefulness of the specific learnings about the site, but in introducing their audience to new perspectives and different modes of knowing. Anderson suggests for example that ethnography has long been valued for familiarizing the strange, an epistemic strategy that has allowed leading its audience to see its own cultural practices in novel light through a newly acquired analytical perspective.

Ethnomethodology has also brought a respect for local meanings to its study of a setting, but on slightly different theoretical premises: it has been committed to challenging the notion that orderliness of human practices is the product of constraining social structures, and in line with this, it has set out to show how the orderliness of a social setting is an ongoing local achievement produced in and through everyday

activities and social interactions. Proponents have repeatedly used this insight to argue against widespread contrary assumptions, notably the idea underlying the project of computer automation that human activity was governed by rules, which could be directly translated into the algorithmic logic of computing (Suchman, 1987; Button, 2000). Beyond this general learning point, it has been suggested that ethnomethodology can provide design with accounts of how orderliness is achieved, notably by describing the practices that make work accountable to rules (Button, 2000). In Figure 8.1, I have emphasized ethnomethodology's theoretical starting point for approaching the social context, which also plays into the kind of theoretical analysis that is pursued.

The third theoretically oriented approach that I would like to discuss here is Computer-Supported Cooperative Work, and more specifically the theoretically motivated programme that Bannon and Schmidt (1989; 1992) have outlined for this area of research. Bannon and Schmidt (1992) have described a theoretically motivated focus on cooperation as interdependent work, derived from the sociological tradition (notably from the work of Marx). On the basis of previous research in the area, they also suggested that cooperative work was a distributed social activity without the omniscient perspective of a central authority, and coordination in this situation had to be local, taking the form of articulation work. Research on information spaces (Bowker & Star, 1998; 2000; Bannon & Bødker, 1997), and the role of infrastructures (Star & Ruhleder, 1996) and boundary objects (Star & Griesemer, 1989) within them, has also been aligned with the above framing of CSCW.

Bannon and Schmidt also insisted on an empirical nexus of CSCW to design:

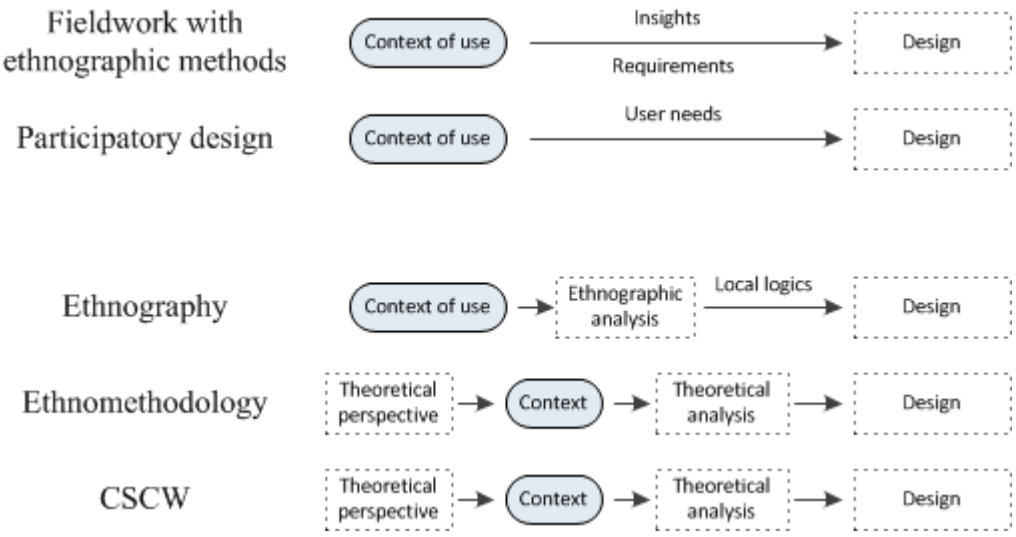
> "CSCW should be conceived as an endeavour to understand the nature and characteristics of cooperative work with the objective of designing adequate computer-based technologies" (Bannon & Schmidt, 1989)
>
> "CSCW is basically a design-oriented research area. […] Thus, the objective of social science contributions to CSCW should [be] […] to explore exactly how insights springing from studies of cooperative work relations might be applied and exploited in the design of useful CSCW systems. This demand not only raises the issue of how to utilize insights already achieved in related fields to influence the design process. It raises more fundamental issues such as: Which are the pertinent questions being pursued in field studies and evaluations for the findings to be of utility to designers? And how are the findings to be conceptualized? If CSCW is to be taken seriously, the basic approach of CSCW research should not be descriptive but constructive." (Schmidt & Bannon, 1992)

Overall, the authors suggested that social science contributions to CSCW should engage in new field studies and evaluations, and channel their findings in a manner that is constructive for design. Unlike Anderson for ethnography and Button for ethnomethodology, they insisted that social researchers should get involved in software design, providing actionable insights, constructing systems and doing evaluations. Despite this shift in emphasis, the importance of empirically grounded contributions was upheld: Schmidt and Bannon basically outlined a programme of conceptually motivated empirical research, where findings subsequently became conceptually reworked with an eye to design. Much research within CSCW was defining itself in terms of the theoretical

problems discussed previously, and it was eclectic in its empirical approach, using whatever method of investigation was appropriate for the problem at hand.

Figure 8.1 is an overview of the relationships that the different social approaches have outlined in connection with the design of software. It serves to illustrate the point that despite making space for a theoretical heritage from the social sciences, social theoretical approaches have been repeating the epistemic stance of the earlier usability and utility-oriented approaches. This stance was also conspicuous within ethnographic fieldwork, which positions social science as the provider of empirically grounded insights of an external context of use. I have used dotted lines to represent the epistemic contexts, such as theory-building and design. The solid line vignettes for the contexts of use are meant to underline their difference from epistemic contexts as settings "in the world". Arrows represent the empirical nexus, marking the epistemic contents that the various social approaches use to mediate the contexts of use.

Figure 8.1: The empirical nexus of social approaches to the design of software



The three theoretically oriented approaches reserve a slightly different role to theory. For the purpose of the present discussion I have set aside the initial theoretical choice of doing social science of some sort, which for these approaches also involves a commitment to engaging with meanings. Ethnography comes to the setting the most theory-free of all three, and it suggests an approach of conceptual construction that emerges from the field. It proposes to serve up to design the theoretically reworked local logics which are operational in a cultural context. Ethnomethodology brings a heavier theoretical baggage to the setting, which actually requires it to operate in a manner similar to ethnography, and to be attentive to the details of local contexts. In its contribution to software design it proposes to combine its theoretical interest in the local achievement of orderliness with the rich detail of the field. Finally, theory in CSCW brings a specific focus on cooperation and its theoretically relevant characteristics; in accordance with this focus, analysis is also aligned with theoretically motivated problems. The suggestion is that the outcome of this analysis should be such that it can drive or inform design; there is also a new emphasis on evaluation, which frames the outcome of design as a context of use to be studied on its own right.

The empirical nexus has also been noted and questioned within social approaches. In Implications for Design, Dourish (2006) describes a social and epistemic constellation ("the politics of design" and "the politics of representation") which configures social contributions outside of the actual terrain of design. While the social-organizational framing within the overall design process is significant, more important for my point is Dourish's suggestion that social approaches are overly invested in practices of representation. Dourish sides with Anderson, Button and Ackerman over the critique of

atheoretical uses of ethnographic methods in fieldwork, and emphasizes the conceptual, reflexive contributions of theoretically oriented social sciences. He specifically speaks up against the expectation that the role of this kind of research should be limited to making recommendations to design, serving up epistemic contents that mediate the contexts of use to the context of designing. He argues that the real merit of these approaches is in their analytic, conceptual work:

> "In reducing ethnography to a toolbox of methods for extracting data from settings, however, the methodological view marginalizes or obscures the theoretical and analytic components of ethnographic analysis." (p. 543.)
>
> "In fact, even in cases where such recommendations [i.e. implications for design] can be concisely and effectively formulated, to focus on those recommendations as the "outcomes" of ethnography at best distracts from, and often completely obscures, the analytic and conceptual work that lies behind, which is *often where the substantive intellectual achievement is to be found*." (p. 547. emphasis is mine)

Dourish's account of the field brings up points that I have addressed previously. In particular, he suggests that the substantive intellectual achievement of social contributions to design should be analytic in nature, but there is a prevalent methodological view within HCI which obscures the analytic potential of sociology.

In a recent book by Dourish and Bell (2012), the analytic import of social sciences is extended in the direction of a conceptual imagination:

> "We would argue that the call for implications for design, drawing on the notion of requirements in traditional software engineering, is a request for empiricism. It

is a request that the ethnography provide " facts " – when people work, how they talk to each other, what they do when they sit down at the computer, and so forth – which can be translated into technological constraints and opportunities. Certainly, many ethnographic studies can offer such things (although it is important not to ignore the role of the ethnographer as interpreter and framer of these "facts," rather than as a passive mirror of the site).

What has traditionally been more complicated has been to establish a deeper, more foundational connection between ethnography and design – to look for a link at an analytic level versus simply an empirical one. The analytic contributions tend not to be seen as holding implications in the same way.

It is not that these do not have profound implications for design, because as we have seen, they do […]. Their impact, though, is frequently more diffuse. *They provide us with new ways of imagining the relationship between people and technology*." (p. 86)

"What we have presented here [in the book] are in fact *acts of reimagining*. […] [M]ore generally, we are arguing that the movement from ethnographic engagement to design practice is inherently a conceptual and imaginative move, not a rote translation of empirical evidence into designed fact." (p. 87)

Dourish and Bell would like to see a reconfiguration of the process of design which incorporates these conceptual modes of working into the design setting. They suggest for example that ethnographies of use are becoming increasingly relevant for design in an era where everyday life has become saturated with devices (Dourish & Bell, 2011). Their account of ubiquitous computing further introduces a significant rhetorical

move which negates the separation of a social reality outside of the realm of technology; the social and the technical become copied onto each other, which is suggestive of a parallel shift where contexts of use merge with contexts of design. As they argue explicitly in the book:

> "[Analytical contributions of the social approach] draw in general on the fundamental repudiation of a traditional separation between designer and user, between technology and practice. To the extent that these implications are not formulated as implications for design, it is because the categories of design, user, and designer are themselves in question."

On this view, design takes place as people enter into meaningful engagement with technology in everyday contexts, and they reimagine this engagement in terms of their understandings of everyday experience.

Overall, Dourish has made two significant rhetorical moves toward an epistemic reconfiguration of social approaches to design:

1. one concerns the shift in emphasis from the epistemic stance of representation to the epistemic stance of analysis and conceptual construction;
2. the other concerns the notion that design is not an autonomous realm overlooking a social world out there: epistemic contexts have collapsed and everything is one continuous flow of social meanings in a world of ubiquitous technology.

These shifts point toward the necessity to significantly rethink social contributions to the design of software. Meanwhile, Dourish has not provided a coherent account of socially or ethnographically based software design which answers to these general calls.

In this chapter, my goal is to outline an analysis which describes a possible model for thinking about the problems of representation and conceptual construction that Human-Centered Computing have been struggling with. The model that I propose focuses on the process of meaning-making at the center of design, which mediates between conceptual construction and learning about the contexts of use. I will also suggest that learning should be seen as an epistemic strategy relying on a distributed cognitive process, which is connected to but also separate from conceptual modeling. This process ultimately draws from experiences of the real world, but also applies various epistemic, conceptual strategies for reworking primary experiences. These experiences involve memory, which can be personal and external. Perhaps even more importantly, they also involve the sharing of understandings. Memory and sharing support a distributed process where the epistemic "stances" of knowing and imagining are in constant interplay. Most importantly, sense-making processes in design frequently prompt the revisiting of knowledge about the contexts of use, on a variable scale. The empirical approach is only one of several strategies that are implicated in knowing the context for the purposes of design. Building a pool of understandings that is available to draw from in conceptual construction may be cited as another major strategy, with underpinnings in a DCog perspective.

8.2. Introduction to case studies

I have previously described how the open-ended framing of the design space prompted participants to construct strategies for finding and constructing meaning. These strategies were applied parallel to engaging in a spontaneous process of conceptual modeling with thought experiments. In chapter 6, I have described the strategies

associated with the practices of prototyping and the use of professional methods. I will now turn to the strategy of pulling in knowledge about the contexts of use as a means of supporting an understanding of the new Sakai. The first case study will outline various epistemic strategies in which the contexts of use were visited, some explicitly derived from professional methodologies, and I will show how the processes of conceptual modeling were relying on knowledge about the contexts of use. Related to this, I will emphasize the mediating role of the knower in bringing domain knowledge to bear on design and innovation. In the second, shorter case study I will suggest that taking the strategic nature of domain-driven innovation seriously requires a cognitively distributed account, which acknowledges the construction of a distributed memory of understandings about a domain, and considers the conceptual contribution of the knowers in connection with this distributed memory. This implies allowing for diverse configurations in how understandings of the "real world" inform conceptual construction, where the practices of building domain knowledge can be expected to coalesce at various moments with practices of conceptual construction, but may also run parallel to and separate from them.

The design space emerging from the Sakai 3 vision invited participants to make sense of web 2.0 for the purpose of higher education. The space was referencing existing examples of software: the abstract formulations of the web 2.0 phenomenon, and the concrete examples of existing implementations out there, in the world at large. The challenge was the translation of this overall approach to the organizational context of higher education. Web 2.0 technologies were perceived as constitutive of modes of online communication and collaboration, and the challenge of translation was approached as the careful insertion of these general purpose technologies into the rule-governed institutional

context of education. This involved understanding how the higher educational contexts of use may inform these general purpose social technologies, and more specifically, finding meaningful ways of mapping the specific organizational and technical realities of the context of higher education onto the web 2.0 approaches.

8.3. First case study: Conceptually grounded collections

Sakai participants had been aware from the beginning that grouping people is central in the day-to-day activity of higher education, and the overarching, organizationally derived educational roles need to be supplemented with various forms of locally initiated notions of grouping. I have suggested that this understanding was playing into the discontent with the rigid structure of sites in Sakai 2. The possibility of creating sections, project groups and other student subgroups within a course, and the ease of supporting non-course based collaboration were important drivers for embarking on the construction of a new platform. It was also noted that social networking allows users to flexibly create and organize their own groups, while the semantic web supports a flexible form of grouping through tagging. At the same time, the problem of groups was set aside in favor of content in early development on account of its complexity, and this resulted in spontaneous attempts at making sense of the notion of groups for the purpose of Sakai 3. The Groups project was conceived as a design exploration focusing on the problem of groups, preparing the ground for future work on the side of the 3akai project.

The initial plan of the project envisaged a series of activities arranged in phases that would be feeding into each other. Thus, the design of mockups was to be informed by the description of contextual scenarios and implementation examples, in turn preceded by a phase of conceptual stock-taking and organization:

> "1st iteration [is] about capturing what we know about groups, making sense of it and putting [it] into forms we can work with (group types spreadsheet, use case matrix, scenarios, etc.)
>
> 2nd iteration is focused on benchmarking and documenting a wide range of context scenarios around creating groups and adding members to groups.
>
> 3rd iteration is design studio which is kick off for wireframing."[124]

Further phases were previewed, with various rounds of prototyping and user testing, but the project did not get to tackle these.

The initial agenda of the project was informed by user-centered methodologies. Daphne Ogle, who was one of the pioneers of user-centered methods within the Sakai community, spearheaded the Groups initiative, outlining and kickstarting many of its activities. Her efforts brought together an enthusiastic group of contributors, many of whom were however not trained in these methods, and often took the project off the proposed tracks. Most importantly, the need for sense-making was apparent across the various activities and discussions, and it was driving initiatives beyond the methods-based agenda. Thus, user-centered design provided the overall motivation and framing, but did not account for the conceptual exploration that was taking place in the web of discourse.

Throughout the project, participants engaged in a series of collection efforts, which were geared toward pulling in relevant details about the contexts of use:

1. A glossary of terms related to groups.
2. A spreadsheet outlining group types in higher education.
3. Interviews with community members about groups.

4. Benchmarking existing interfaces.

5. A description of sources of group-related data in the broader software ecosystem of higher education.

6. Descriptions of mental models related to groups.

7. Contextual scenarios.

As I will show in detail below, the collections were operating with a wide array of epistemic strategies for making knowledge of the context available for the purpose of design. They could rely on participants' own preliminary knowledge, sometimes in combination with imagination, or on the knowledge of others. In either cases, the collections were always connected to communication and sharing, and the desire for conceptual clarity was coupled with the desire of sharing understandings and building common ground. Some were focusing on the educational context, others were more generic. Some were focusing on the interface to be built, some on an existing socio-technical ecosystem of software, others were technology-agnostic. These conceptual boundaries were not strictly maintained, and collections operated with a heterogeneous mix of perspectives.

Going further, I will show that the collection of details about the contexts of use was not only driven by collaborative sense-making, it was also sparking spontaneous attempts at conceptual modeling.

8.3.1. A glossary

The idea of creating a glossary came up in discussions preparing the launch event of the Group project. Subtle differences in terminology were quickly registered in these very early discussions, and a suggestion was made to focus on the "semantics." Daphne Ogle

followed up on the suggestion, and kickstarted the very first collection effort in the project with a number of entries and definitions. In the publicity for the effort, the goal of creating a shared language was emphasized:

> "It's just semantics but having a common language will go a long way in effectively gaining a shared understanding of the problem space and project."[125]
>
> "I started a glossary that we should all continue to build out. … This will help us have a shared language... or at least know what each other are talking about if we don't :)."[126]

At least three other people contributed with definitions[127].

> The items covered in the glossary were:
>
> Collaboration space, Community, Course Cohort, Course Site, Instructor (Also referred to as Faculty, lecturer), Grader (They may or may not also be a TA), Parent Group, Permission, Provisioned, Relationship, Role, Subgroup, parent group, Subject Coordinator, Subsite, TA (Also referred to as GSI).

The list itself was heterogeneous insofar as it contained terms specific to education (faculty, grader, TA) and others that had a broader usage (community, role), as well as terms from within the software world (permission, provisioned). The educational and software domains both influenced the description of general terms with broader usage. The only context-free definition was that for community:

> **"Community**: A group of people who share a common interest, residence, occupation, workplace, etc. A clear distinction is notoriously difficult [hyperlink to http://www.google.com/search?q=Bloom+A+nation+is+the+same+people+

living+in+the+same+place], but most people probably think of a "community" as being larger than just a "group", and when we select a "team" (or some other set of people) it's generally from some communal context."

This characterization also contained an associative reference to a quote from James Joyce's Ulysses, which evokes a scene where the definition of the nation, one possible example of community, is being discussed. The quote was meant to illustrate the difficulty of grasping the concept.

Educational and software-related definitions were also heterogeneous in the sense that some descriptions were referring to a technology-agnostic social world, others to an institutionally-agnostic software world, and yet others to an educational socio-technical, world at the convergence of the two. The different perspectives were also applied subsequently in a single description. Thus, group was described first in general, and subsequently with reference to a software system:

"**Group**: A collection of people, groups, or some combination of the two. Groups may persist or be dynamic. Dynamic groups may be constructed ad hoc by some query, generated by either the system or some user action, and are transient."

Course Cohort was similarly first defined as an educational term, and subsequently nuanced in terms of the socio-technical practices of a School Information System (SIS):

"**Course Cohort**: An externally defined group whose manager is an instructor of record, and which may comprise other externally-defined groups as sections. Course Cohorts generally have additional data or restrictions demanded by the SIS. Also referred to as: **Lecture, Course, Class**, **Primary Section**"

Other concepts were unlike the previous examples, in that they were based on the existence of a software-based socio-technical system. The definition of Course Site, in particular, supports my earlier point that the design of Sakai 3 was approached as the insertion of general purpose software exemplars into the institutional context of higher education:

> **Course Site:** Two aspects distinguish a course site from other online collaborative spaces: (A) Incorporation of (or integration with) existing academic social structures, often basing online access rights and user roles on things like department position, cohort membership, class enrollments, section assignments, tutorial groups, librarian specialties, and so on. (B) Domain-specific functionality: assignments, grading, use of mentor-like roles, and so on.

A course site was distinguished here from the generic exemplar of "online collaborative spaces" on the basis of the incorporation of "social structures" (like roles, classes and organizational positions) and educationally relevant functionality (like assignments and grading).

As the examples illustrate, the glossary was not really an attempt at providing definitions. It was rather crafting descriptions for the purpose of creating a common ground in collaboration. The descriptions were filled with associations that were placed there to shed light on the usage and usefulness of the terms. Thus, the inclusion of dynamic and persistent groups "defined" the group by pointing out that both of the two complementary group types fell under the definition of group. Contributors were hopping between social, educational and software contexts with ease, making conceptual

connections across the various perspectives, which could even include works of literature.

The descriptions were also referencing a dimension of variation within the educational context, and the act of construction was also evident in this respect. Thus, variants were listed for some of the terms (i.e. Course cohort – "Also referred to as: **Lecture, Course, Class**, **Primary Section**"), or local ways were spelled out, as in the case of section:

> **"Section**: A real-world subgroup for a course. In some universities, courses can be split into several large lecture sections each term, with smaller subgroups meeting in discussion, lab, or studio sections. At UC Berkeley, official student enrollments and final grades are tied to lecture sections (which are thus called "primary sections"), but practices and terminology vary widely across institutions."

Contributors were clearly aware of and acknowledging the detail and heterogeneity of institutional realities, and their goal was to construct a conceptual umbrella that would allow for rising above them, and to think above and across these differences.

8.3.2. A spreadsheet outlining group types in higher education.

The group types spreadsheet[128] was a Google spreadsheet set up by Daphne Ogle for collecting descriptions of all kinds of groups in higher education. She described it as an effort "meant to help us get us understand this space [i.e. the conceptual space around groups]. It represents concrete examples of types of groups, how those groups get created, who is involved, typical interactions, etc." The purpose was to grasp different

conceptual logics underlying the notion of groups in higher education by means of capturing concrete examples.

The spreadsheet contained a listing of groupings in the rows, with an example (see Table 8.1). Each entry could be described from a number of suggested perspectives given initially by Daphne Ogle in the columns. These included both platform-independent and platform-related descriptions:

Table 8.1: Examples of perspectives given as columns for describing group types in the spreadsheet

Platform-independent	Platform-related
Typical length Dynamic or Static Actors in group Subset of Type of interaction / Main goals	Manual or Automatic creation Actors that interact with group Where does membership come from? (externally provided, manually self-enrolled, manually enrolled by other) Collaboration Space / Needs Attributes/metadata about group Attributes/metadata about members of group Contexts likely created in

The effort drew contributions from at least four other contributors.[129] Contributors were filling out the spreadsheet based on their personal knowledge of the world of higher education, pulling up group-related conceptualizations from their memory along the lines suggested by the framing of the document. The different types of groups were also

arranged under various categories[130], which, together with the examples already provided, suggested a generative logic for constructing new examples of group types, or new categories. Thus, looking at the listing of "Officialish" groupings in Table 8.2, we see groupings around different roles within a same class (everyone, instructional stuff, instructors of record, head TAs), and the extension of this logic to same *type* of class, covering all the different instances ("sessions") of its provision (as exemplified by Spanish 1-001 versus Spanish for 1-001, 002, 003, 004). Similarly, the last category presented in Table 8.2. plays with the notion of dynamically pulling together groups based on criteria that the system makes available as data. This last category appears to have been created in an attempt to expand the conceptual space and go beyond the logics that had been outlined, as suggested by the doubts expressed in the explanation: "is this a true group?", "is this a search result?".

Meanwhile, there was no attempt at seeking comprehensive coverage and creating conceptual order beyond these generative headings. The entries were accepted to be heterogeneous. Thus, "officialish and "student groups" generally represented groupings that were part of the organizational arrangements of education, while "All guests with expiring accounts", or "Students who have viewed this content" would only exist within the system, and "Randomly made student groups" and "Who's nearby" assumed some form of technology for displaying groups. The miscellaneous character of the last entry, "Misbehaving individuals", also testified to an attempt at overcoming conceptual boundaries.

Table 8.2: Selection of entries from the Group types matrix spreadsheet[131]

"Officialish" groups in Courses	Examples
Everyone in class	Instructor, TA's, students & guests for Spanish 1-001 (several sessions of this class are taught)
Instructional staff for class	Instructors, TA's, guest lecturers for Spanish 1-001 (many teach sessions of the same class)
Instructors of Record	Instructors responsible for signing the gradesheet submitted for a class and being evaluated as main instructor for a course or section during course evaluation process
All instructors teaching the same class	All instructors for Spanish 1-001, 002, 003, 004...
All TAs for X class	TAs for Spanish 1-001
Head TAs	Chemistry classes have a Head TA role who manages the TA and Grader team
Student project groups	
Randomly made student groups	Class of 100 and instructor wants students in random groups of 3
Feedback teams	Peers review work that gets incorporated in final grade.
Student lab group for assignment	Short term collaboration for an assignment. Turn in 1 copy.
Problem-based learning student groups	collaborate on a specific problem, perhaps present
Vertically integrated project teams	Collaborate on complex multi-year problems in engineering (dynamic group from all levels)
Capstone Experiences - small student group across semesters	collaborate on deliverables
Common interest /affiliation	
Friends in X class	is this a true group?
Who's near-by (location aware)	Want to meet up for dinner
All guests with expiring accounts	is this a true group?
Students that live close to me	isn't this a search result?
Students who have viewed this	is this a true group?

content	
Students currently viewing this content	is this a true group?
Anyone who's even taken chem1a	isn't this a search result?
Students who haven't completed X	is this a true group?
Membership of student societies and clubs	social activism, sports club, Dining club, academic interest, fraternity
Misbehaving individuals	Is this a group?

8.3.3. Mental models collection

The spreadsheet and the glossary initiatives suggest that participants were thinking of the design challenge around groups as inherently conceptual, where they were tapping into their understandings of the conceptual space around groups to gain a better sense of the role of groups within a new Sakai. While this orientation remained implicit in the first two collection efforts presented above, the mental model collection was explicitly framed as an exercise in reviewing mental structures around groups. As the heading suggested:

> "The exercise was meant to get ideas our of [out of our] heads and onto paper about user expectations, good interactions we've encountered around groups and any other relevant models."[132]

The collection included two threads: a collective brainstorming session, which was transcribed and shared in the Confluence, and a presentation about a personal model of groups, which was followed by a discussion. Further presentations of personal models were planned, but these did not take place.

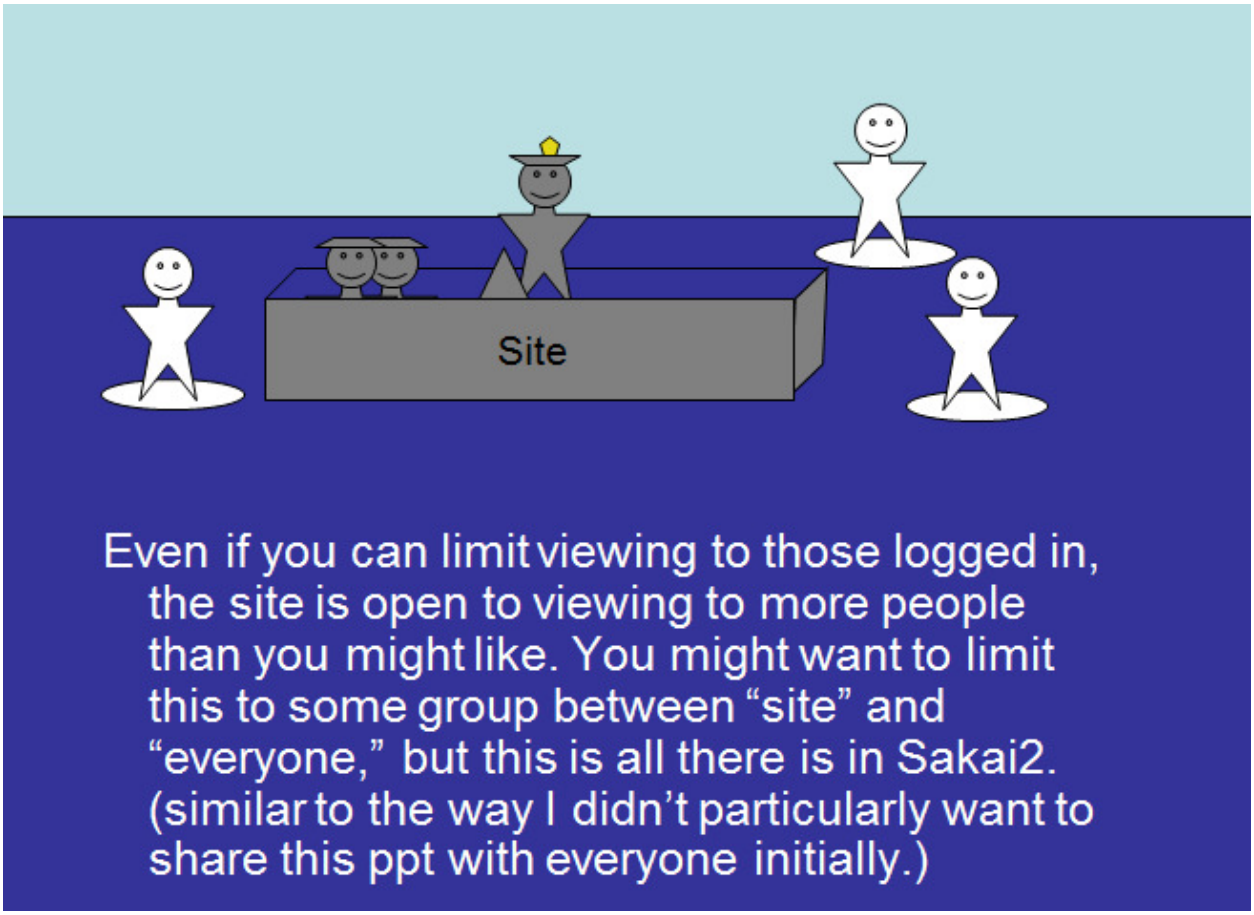
Overall, the mental models surfaced in the brainstorming were similar to the collections described previously, featuring a grab bag of heterogeneous perspectives

around groups. They are interesting for us now because they show how the conceptual collections about the state of the world became spontaneously meshed with mental simulations and conceptual construction. Thus, Keli Amann's presentation was entitled "A mental model of sites and groups in Sakai 2 and 3", which suggested that it would be spanning the existing and a future system, and it started with the following "Disclaimer":

> "The following is one person's mental model of groups and sites and how they are related. It is like an analogy – it is not a representation of the entities of groups and sites as a developer might describe them. They are also not representative of the thoughts of all those working on this project and may not be accurate. This is intended to spark conversation and could either be expanded on or picked apart."

The conceptual language was vague, but the author clearly made an effort to set her contribution apart from the approach which seeks *representativeness* among stakeholders, and an *accurate representation of entities* in the real world. Instead, it was framed as a *personal* account, based on *analogy*, and intended to *spark conversation* and to be *expanded*.

Figure 8.2: In Sakai 2's Waterworld, sites are like ships, and individuals have no homeland, save for a tiny board



The presentation started with the analogy describing Sakai 2:

"In this world, sites are like ships, and individuals have no homeland, save for a tiny board. It's like 'Waterworld'."

Figure 8.2 shows one of the illustrations for Sakai 2's "Waterworld". The narrative continued with outlining a thought-experiment, where "an individual comes along and wants to look at your site, he has two options…" The two options were in turn described and their implications were teased out:

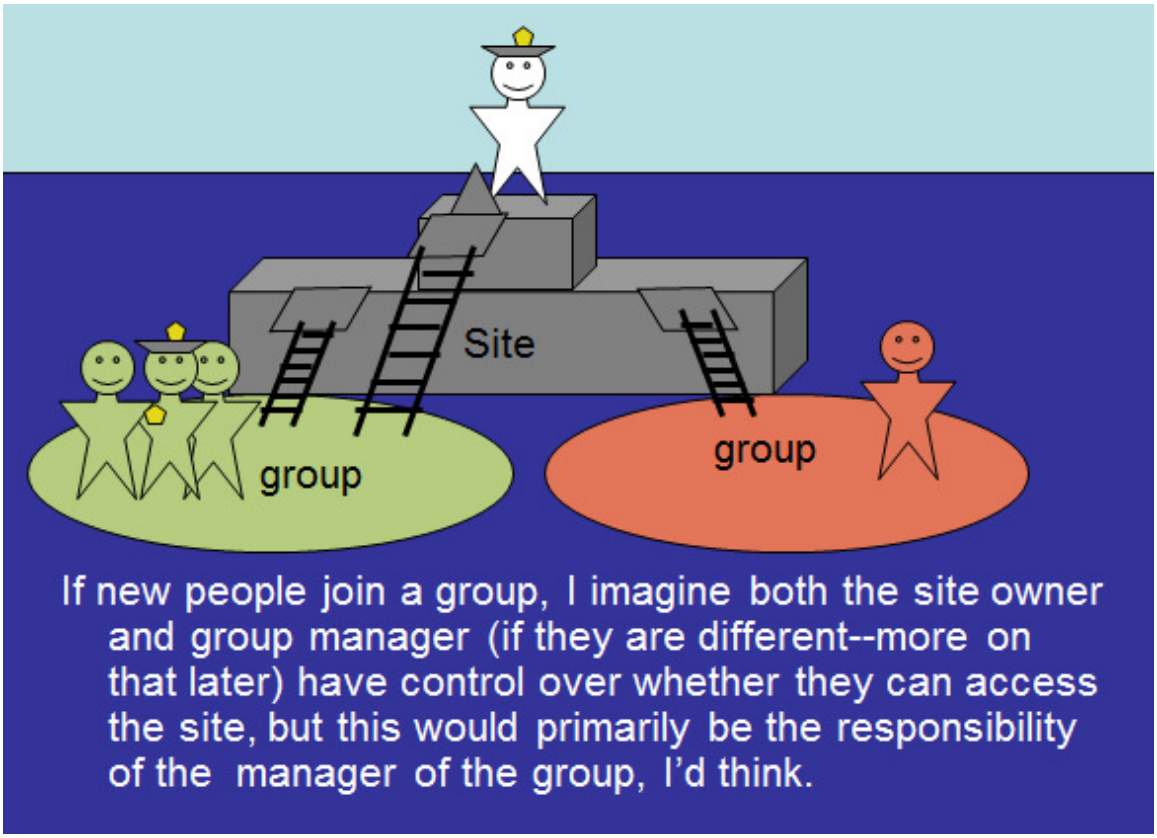
> "One, he can go through the process of joining the crew and becoming a member of the site. That's great if he really wants to participate and get all the announcements, but if he just wanted to see the content in the site, it's a bit more than he needs."

The narrative suggested that the implications for the first option were that joining a crew would result in getting all the announcements, which could be too much information. The second option described a solution to this problem, in which the manager set up a site where content was public and open for everyone to see. Again, the tale continued with the implications of this choice. A second thought experiment was also outlined:

> "In this world, if you have a new site, you have to individually move and reenlist everyone."

The presentation did not follow through with this second thought experiment, only indicated a direction for doing this in the form of the bracketed question: "what are the disadvantages?"

Figure 8.3: Sakai 3's Waterworld, where individuals live on the floating islands of groups, and they can visit content-ships



In contrast to Sakai 2, Sakai 3 was described as a Waterworld where "Groups are like floating islands – they can exist autonomously from sites." (See as well the illustration in Figure 8.3.) From here, the narrative followed the same logic, describing the situation as a thought experiment, and teasing out the implications. This time, it was a possible rather than an existing solution that was outlined. The suggested solutions opened up further questions, which prompted further conceptual explorations, explicitly labelled by the narrator as "thought experiments":

> "How does a site or a group get started? Does one have to come before the other? Here are a few thought experiments…"

This second series of thought experiments played around with starting from groups and starting from sites. One of the implications of the suggested model stated that "[g]roups do not exist as an end to themselves – they exist to selectively manage access to information." This was however further qualified in the concluding slide:

> "So how might content be depicted in this model? Bear with me as I stretch this metaphor [sic! What had been referred to as analogy now became a metaphor.]… Imagine a thick, soupy fog has swept in and lifted the content off the site. It's still visible to all those in the site and the site owners can still manipulate it. You can only see the content if you have access to the site, but I don't believe the content has any direct relationship to the group. If that's true, we don't have to think about content at all – all we have to worry about is the relationship of groups and sites."

It was thus suggested that the conclusion of the tale of Sakai 3's Waterworld was a structural setup, where groups were connected to sites, and individuals could see a site's contents on account of being members of a group, so content would be made accessible on account of being part of a site. Thus, the thought experiments were built around the guiding notion that groups and sites are primary constituents for managing access to information.

The various conceptual collections that I have surveyed so far approached the problem of groups with generalizing conceptual abstractions. They were tapping into the knowledge of participants about the social or socio-technical domain of education on this plane. Meanwhile, the brainstorming and the conceptual tale of Sakai's Waterworlds prompted the emergence of a conceptual playground where participants were putting the

suggested models to the test of their own understandings, and in their arguments they also tapped into the particulars of the world of education.

Comments were continuing in the spirit of thought experiments, and a common strategy for thinking through and testing the suggested generalizing models was to pit them against specific situations. Daphne Ogle in her comment to Keli Amann's mental model presentation was pointing out a distinction between group membership in the real world and in a system, which she attempted to capture by the concrete scenario of inviting a person to visit the website of the Sierra Club, so that they could learn about the club before they would decide to pay the dues required for becoming a member:

> "I think it means something different to join a group than to join a site. When I join a group there are relationships involved which you refer to earlier. […] So I don't think they should be the same or are the same kind of "group" necessarily. In fact, I could see inviting someone to join a site (create an account on the Sierra Clubs website so you can access the newsletter etc.) but that person hasn't actually become a Sierra Club member until you pay dues. In fact a large motive for the site itself might be to get you to become a member." [133]

While the Sierra Club example came from the world at large, examples of specifics would typically be drawn from the educational domain. In the following comment, Ray Davis argued that there should be a conceptual distinction between group and membership list, and he gave snapshots of specific examples from the world of higher education where membership list made more sense than group:

> "The unqualified word "group" might not be enough to clearly guide our designs. The functional UX space we're targeting is "managed and/or targetable

membership lists." Those membership lists sometimes belong to things that our users don't explicitly think of as "groups" because their group-ish aspects are conceptually overwhelmed by their other aspects. I might want to add members to the "3akai Design Workspace" or to see members of "Balliol College" or to send email to "Psych 202" without having to deal with a superfluous "group".[134]

In a subsequent comment, Ray Davis outlined four different approaches to external membership feeds, illustrated with more detailed scenarios from higher education. One of the examples described the workaround for overriding externally provided memberships, and the information management challenges that the workaround implied:

> "E.g., an instructor wants to add a waitlisted student and a teaching assistant to her class, but the SIS [Student Information System] organization which controls the external feed isn't set up to cope with those less-official cases. And so the instructor gives them non-SIS-controlled access to the class's online workspace just as she gives them non-SIS-controlled access to the class's physical meetings. This requires some (hopefully well-understood) form of reconciliation between external feeds and online overrides. If I add a student unofficially to a class on the assumption that he will soon show up in the external feed, and he does show up in the external feed, and he then drops the course, I don't want to keep seeing him as an online member of the class." [135]

In the discussion after the mental model presentation, Ray Davis was requested to explain a conceptually abstract comment he made, and he responded by spelling out concrete examples in explanation:

> "It's never hard to get Ray to expound.

> In email and discussion yesterday afternoon, I noted that most (not all) collaborative workspaces have roles to distinguish one kind of member from another, but that these roles do *not* rigorously map to the *outside* of collaborative workspaces. (There's no standard answer to the question "Is a Chem 10a student a discussion forum moderator?") So when we talk about roles, we always need to be clear about what context we're in."[136]

The various contexts cited were spanning education in general, existing web-based systems and existing implementations of educational platforms:

Example from (higher) education at large:

> "Most classroom contexts probably include the roles of "teacher" and "learner," but many instructional contexts have more or fewer." [137]

Example from existing web-based platforms:

> "Many collaborative workspaces get by with just one type of member – every member has the same rights and gets the same UX – and so they can also get by without explicit roles (although even they may assume an implied special role of "owner" or "adminstrator")."[138]

Example from the 3akai implementation:

> "The current wiki-like functionality of the 3akai demo defines three member roles, and when you add members to your site, you have to say what role they play there: Viewer, Collaborator, or "owner/admin" (which isn't visible in the management page at present but which certainly acts as a role)." [139]

Example from an educational platform unrelated to Sakai:

> "As it turns out, the fairly rich functionality of Harvard's iSites / iCommons suite *also* gets by with just three "permissions" called "View", "Participate", and (again, mostly hidden) owner/admin." [140]

Example from the web at large:

> "Keli pointed out in email that Google Docs *also* has three roles: Collaborators, Viewers, and Owners. Obviously this trio can cover a lot of territory." [141]
>
> "However, roles/permissions at Google Groups already start to get more complicated. And we know that certain applications may need to openly deal with more than two types of person […]"[142]

Example from the Sakai 2 platform:

> "[…] (here at UC Berkeley, our gradebooks have people who can add assignments, people who can give course grades, people who can grade only a subgroup of students, and people who get graded)." [143]

The examples I have cited contained snapshot-like visits to the specifics, but the practical anchoring of conceptual models could be more expanded and more detailed. For a good example of this, I will move outside of the Groups project, to a design review session, where Clay Fenlason was commenting on the suggested separation of site setup from adding members:

> "I think you've got two basic halves to site setup: one is setting up the structure of the shared space itself (along with any initial content), and one is joining people to it in appropriate roles. I think we'll be better off if we treat that second region as its own area, optional for initial site creation. […]

> Let me attach a use case[i]: […]
>
> A professor is going on sabbatical for a semester, but knows that he's scheduled to co-teach a seminar course starting in January with a colleague. Since he knows he'll be moving back in December, and preoccupied with both that and the holidays, he suggests that they get the site set up now and start working on it gradually while he's away. It's a new course, and it doesn't have a registration code or official title yet - that paperwork is going through the department and it will get settled at some future point. But those are details that can wait: right now he just wants to set up his course site so that over the next few months he and his colleague can collaboratively winnow through the material which will end up comprising the course."[144]

The examples above suggest that in the activity of conceptual exploration, participants were gliding seamlessly between generic and more specific conceptualizations of phenomena, and they were tapping into their memory for practical, concrete details to support the efforts of model construction. The conceptual models at various levels of generality and abstraction may be described as containing conceptual handles to the context in which they operated. These handles could be opened up, so there *was* a possibility of unfolding their reference, but this was not required for engaging with the models, which could easily function abstracted from the context. In the above examples, conceptual models were expanded with the purpose of finding out their implications or explaining and illustrating a fine point to others. This is a movement from the generic to the specific, but the overall flow of the conceptual arguments suggest that

[i] The cited use case is the second of two described by Clay Fenlason.

the reverse direction was equally practicable; more precisely, participants were navigating a conceptual web, which spanned various levels of detail and perspectives. This web was anchored in participants' long-term experience with educational software technology.

As I have shown earlier, participants were engaging in the various conceptual collection efforts with the idea of making sense of groups for the future Sakai platform. Envisioning the role of groups within Sakai 3 primarily involved tapping into and aligning the conceptual models of the contributors, and the specifics were invoked as part of this process. Thus, the various levels of conceptual understandings were visited from the perspective of conceptual modeling, which was itself framed in terms of designing interactions with a user interface. The UX-perspective was prevalent in the different conceptual construction episodes cited previously. Keli Amann's presentation about the "mental model" of groups in Sakai 2 and Sakai 3 was formulated in terms of engaging with the interface, describing sites that are created, joined and viewed. The ensuing discussion continued along the same lines. In the discussions after the brainstorming session, groups were discussed in connection with joining sites, or managing membership lists. The specifics were pulled in in accordance with this logic, but this did not necessarily mean that the details were talking to user actions with the interface. Sometimes they did, as in Ray Davis' example of the instructor who wants to add a waitlisted student and a teaching assistant to her class, or Clay Fenlason's illustrative use cases, but in other instances they also included technology-agnostic notions about the world of education, with instructor and student roles, classes, colleges and student dorms.

I will close the discussion of conceptual collections with the analysis of a series of comments by John Norman that brought together the various threads presented so far. John Norman responded to the distinction between implicit and explicit groups in the brainstorming session. After pointing out that the notion of implicit groups contradicted the very idea of an interactive interface, he went on to outline how implicit groups may be translated into a meaningful user experience. His speculations show how embroiled ideas about the user interface and the educational world became in conceptual construction.

> "You mention explicit and implicit groups. This is an important distinction which is perhaps not explored enough. I think it is going to be quite hard (impossible?) to manipulate implicit groups, so an important interaction might be to capture an implicit group to make it explicit **or** to be clear that you can't do the same things with an implicit group as you can with an explicit group."[145]

John Norman started here from the conceptual account of a common sense distinction between explicit and implicit groups, which we may assume to be rooted in everyday knowledge of the world. He then applied this distinction in an imagined, conceptual model of the interface, and pointed out an attendant challenge: implicit groups would be implicit in the interface, and thus hard to manipulate as such. From this he drew the implication that implicit groups needed to be made explicit in the interface for the purpose of manipulation, and outlined two conditions that could be explored as thought experiments: accepting that implicit groups would be unavailable in the system, or letting users capture them.

The latter suggestion pointed in the direction of a novel solution: finding intuitive ways of making implicit groups explicit in the system. John Norman continued his train of thought in this spirit, outlining an approach where tagging would allow the capture of some implicit groups.

> "As I write, I speculate that one intermediate step between a set of people and a group (technical) is tagging. Thus, if we think of the set of people who are residents in a college, "Membership" of the set will be controlled outside our application, but if we assemble a list of such people (e.g. by a search of address information) then we may want to tag them with their 'belonging to a college' status."

He started by describing how tagging may be achieved in a practical sense: the source of tags could be a user, or an external database.

> "This would allow us to [do] certain things (like add the college calendar to their homepage) more easily, but will make other things harder (such as removing the tag if they leave the college)."

Then he moved on to outlining a new conceptual model centered on the distinction between tag-based and membership-based grouping in terms of user interactions "in our application" , dressing the model up with details crosscutting various domains of experience:

> "Of course, a set of people with a common tag can easily be turned into a group and membership maintained *in our application*. So some sort of understanding of temporal issues and the need to manipulate membership seems to be important in determining whether we want to handle the set as an explicit group in our

application. E.g. eye colour is unlikely to change, so a set of blue-eyed boys is likely to be stable and not require membership management, so could sensibly be handled by tagging rather than groups. On the other hand, although the set of people registered on a course is not directly under our control, we need to manipulate the set of people who have access to a course so it makes sense for us to make a group for that access and manage the synchronisation of that group membership with a data source that describes the current set membership. Finally, there are sets of people that only have meaning/purpose within our application (e.g. the set of superadmins) where we have a need to manipulate the membership and full control over the information, so a group is clearly the way to go.

Sorry for rambling."

The result was a conceptual model with three types of groupings: a first type of grouping based on (relatively) stable, system-external characteristics, and thus manageable by means of tags; a second type based on system-internal characteristics, represented in terms of group memberships; and a third type, based on temporary data from outside of the system, represented again in terms of group memberships. Examples for the three types were coming from three different domains. The stable grouping was exemplified by the example of blue-eyed people, culled from the world at large. The system-internal grouping was exemplified by superadmins, a generic software engineering concept. Finally, the temporal grouping was exemplified by the educational notion of course registration, and the action of registering students in an educational platform.

Overall, John Norman's "rambling" was outlining a novel conceptual model in terms of meaningful activities with a user interface, and argued for the meaningfulness of

these activities based on quick references to various domains of experience. He gave his arguments with the purpose of convincing other participants in the Groups project, and possibly beyond, to align with his suggested model, and for this, he was relying on specifics that could be expected to be part of the experiences of other Sakai members.

8.3.4. Contextual collections

So far, I have been showing how the goal of making sense of groups in Sakai 3 prompted participants to engage in conceptual collection efforts, for which they were relying on their memory and imagination, and they were lining up their practical experiences of various Sakai-related domains of experience to construct and run thought experiments. The early conceptual explorations of the Groups project were followed by a series of collections, where participants made more sustained forays into the various domains of experience, collecting rich details from the contexts of use, from which their sense-making efforts could expand.

This second layer of collection consisted of a series of a series of initiatives which did not follow an overall plan, but sprang instead from the interest and perceived need of those involved. The federated use case collection and the interviews were performed by a single participant, Ray Davis, while the other two initiatives were collections initiated and set up by Keli Amann. The initiatives were covering various domains:

Table 8.3: Overview of domains covered by the contextual collections

federated use cases	socio-technical context in higher education
interviews with community members	social and socio-technical context in higher education
benchmarking existing interfaces	broader socio-technical context
context scenarios and use cases	human context: user goals and interactions

Federated use cases

Ray Davis set out to chart how existing systems in higher education act as sources for permission and authorization[146]. The collection was motivated by the widespread higher education practice of maintaining external systems with semester- and course-related student registration and employment data, such as a Student Information System (SIS). These systems and the underlying databases had been traditionally separate from teaching and learning oriented platforms, like Sakai 2 (usually referred to as Content and Learning Management System), and acted as external providers for authoritative data on the official status of participants within the educational process. Typically, these systems would be based on high-level categories, such as roles like student and instructor, and instructional units, like curriculum and course offering. The categorizations were mirroring the institutional legal framing within which education operates: financially binding relations, legal expectations about accreditation, and so on. In this quality, they constituted a resource for Sakai 2, which was however often perceived as constraining rather than merely constitutive of participant privileges. As Ray Davis formulated the motivation for his work:

> "A Learning Management System (or Collaborative Learning Environment, or Framework for and Suite of Web Applications Used by Teachers, Students, and Researchers in an Institution of Higher Education) typically needs to deal with far more complex authorization requirements than a hosted web application, a corporate site, or a government site. In higher education, a single individual may play an instructor in a classroom, a student in a seminar, a leader in a discussion group, a grader in a lab section, and an editor on a research project. Some of these roles and contexts (but not all) are typically determined outside the LMS. These context-dependent roles imply different application roles and permissions. The set of possible roles and their implied permissions vary from institution to institution, and even within an institution. The applications (and their own ideas of user functions) are developed independently and deployed over time.
>
> What follows is an initial culling from known requirements for groups, roles, and privileges. Group and role management use cases take the most space, but I also mention less visible authorization integration requirements that have caused QA and maintenance pain over the years."

Besides describing external systems as a source of constraining data on user privileges, Ray Davis also suggested that their categorizations resulted in role-based groups. Implicit in his account was the argument that making sense of groups and the related privileges for the new Sakai required a sense of the institutional context of education, with its grouping practices based on formal and legally binding categories, and more specifically it required the understanding of the mediation of groups by existing socio-technical systems.

One of the accounts described how privacy preferences derived from a legal environment may be mediated to Sakai by external systems:

> "Students may have officially requested non-disclosure of certain information in the systems with which we integrate. The legal requirements of FERPA frequently come up in the United States; typically this might include a rule that official instructors of the class are the only participants who can see certain information about an enrolled student. (Hospitals attached to medical schools may have a different set of stringent requirements set through HIPAA.) [etc.]"

Some were talking to the insufficiency of the categorizations in the external systems, and the challenges of fine-tuning them:

> "Merging external memberships
>
> We frequently need to create project-oriented, subject-matter-oriented, departmental, or cross-listed workspaces whose memberships are drawn from multiple courses or multiple campuses. Naturally, however, we can't accidentally let a teaching assistant in a lecture class grade herself in a seminar. For this and other reasons, it's important to preserve backward links to the original information sources, and to define clear ways to reconcile role-mapping conflicts. (Oliver Heyer)"

Others were describing the peculiarities of institutional realities, like limited spaces in classes, or multi-campus collaboration:

> "Space-limited student sign-up
>
> Each year, about 100 medical students sign up for elective projects (with associated online workspaces) with a supervisor. Each supervisor can only work

with 4 students at most. Some supervisors and topics are much more in demand than others, and so competition can be fierce."

"Merging from multiple external authorities

[…]

Large multi-campus universities or universities which include the option of study abroad may need collaborative spaces which combine memberships from different identity management or course management systems."

For the description, Ray Davis was taking his own professional perspective as an instructional software technologist, positioned at the juncture of day-to-day educational practices and software systems. This perspective implied that he could create composite accounts from the fine details of the situated and goal-driven educational activities and the software-based means of supporting them. Consider how the two perspectives coalesced in the following two accounts:

"The link between online memberships and externally-defined memberships is not *equivalence*. We may want to give online memberships to people who do not officially (or do not yet officially) play a role according to SIS. That means reconciling multiple sources of membership changes. For example, an instructor wants to let officially enrolled students automatically gain access to her class workspace, but doesn't want to punish students who are still working their way through SIS official channels, and also doesn't want to have to monitor the status of every "early" student separately. [etc.]"

Overall, he described a world which was already saturated with technologies that acted in various ways to mediate institutional rules, as well as ensuing local practices of accommodation and workaround.

For outlining these accounts, or what he described as “group and role management use cases”, he was relying on his professional experience. While he clearly appeared knowledgeable with respect to the fine details of educational systems, the actual use cases were often credited to other participants within the Sakai community:

> “The names in parentheses are experts whose accounts I've drawn from and who I hope will correct any mistakes in my summary.”

Some of the descriptions were also referencing external pages with additional information, but this was rather an exception. Given the detail and the wording of the descriptions, it is probable that most uses cases had been originally shared in written form, possibly in email or on Confluence. Some were clearly copy-pasted, as the previously cited use case of “Merging external memberships”, which was attributed to Oliver Heyer and preserved the original 1st person pronouns:

> “We frequently need to create […] we can't accidentally let a teaching assistant […]”.

The author did not hesitate to admit the method of “slavish copying”:

> “The remaining pedagogical use cases are all slavishly copied from Clay Fenlason's list of Georgia Tech Group and Course Requirements.”

Other use cases appeared as compilations, indicating for example several sources, or no source at all. The collection contained close to forty entries, which were carefully organized and in most cases edited or reworked in support of the goals of the author, who

had clearly appropriated the knowledge that had been culled from the community, and made it available so that others could learn from it in a similar manner.

Interviews with community members

The interviews[147] were also carried out and subsequently summarized by Ray Davis. The effort may be seen as complimentary to the collection of accounts previously described by community members, as it was involving them directly in describing group management practices at their institution. Eleven different interviews were made, some focusing on an institution (i.e. the University of Oxford, NYU), a solution (OpenSyllabus, Drupal), or a shared issue (integration, templates). Depending on the interviewees, the actual content may have a technical, data-driven or organizational focus, but the overall approach was similar to that described for the federated use case collection, with a socio-technical perspective combining systems talk with descriptions of day-to-day institutional practices.

Benchmarking existing interfaces

This collection was directed at understanding and describing how existing social media sites (such as Flickr, Facebook, delicious or Google Groups) and role-oriented software systems (like Drupal, Moodle or Atlassian Crowd) deal with groups. Eight participants described about a dozen different sites and systems, each taking full responsibility for one or a few of the cases. The areas to be covered were defined by the initiator, Keli Amann, and included such activities as finding, creating or joining groups. On the Confluence page, she also described the collection in terms of the professional method of Competitive Analysis, for which various authoritative references were provided. Some

accounts ran considerably long, with a large number of illustrative screenshots, others were only a few paragraphs.

The primary perspective of the descriptions was that of the user interface, and the content-based interactions it made possible. In the following excerpt on Google Docs, the available user activities are viewing or editing a file and inviting others to view or edit:

> "Groups are used to determine who has access to view and/or edit a file. Depending on the settings, the group can view the file, edit the file, invite others to view to edit or view the file, and/or send invitations to view or edit a file to mailing lists that work for all mailing list recipients." [148]

The accounts also acknowledged the metaphorical nature of conceiving of user activities. Thus, in the following excerpt about delicious, the author suggests that the site uses special language, like bundles, and networks of fans, to describe phenomena which could be easily translated with activities more commonly associated with group management:

> "There are not really any formal groups in Delicious. When you follow another Delicious user (to borrow a term from Twitter) they become part of your *Network.* Delicious also says that you are a *fan* of theirs, and if they follow you, they are your *fan* but this terminology is inconsistently used. You can also organize different sub-groups of your network into *bundles* so that you can easily see all the bookmarks tagged by your "friends" or the people on the groups project." [149]

The interface was a naturally given standpoint in the case of sites, typically representing the sole form of public access to the running system. Some of the systems, like Moodle, were also approached in this manner. Others were described from a comprehensive standpoint, spanning from database structures to user-facing offerings. Atlassian Crowd

was for example the system behind Sakai's Confluence, and for this reason, it was known from the inside to many participants, so it came to be characterized in software developer terms:

> "A commercial product to manage integrated authentication, personal information, and group memberships across multiple applications including other Atlassian products, Subversion, Google Apps, OpenID-enabled sites, any Spring-Security-based application, and pretty much anything else."[150]

Related to the review of existing systems it may be noted that other educational platforms did not figure prominently on the group's agenda. Proprietary platforms like Blackboard were not included because the community was explicitly limiting itself to open source solutions. Moodle was thus reviewed with respect to groups on this account. Meanwhile, many of the general purpose offerings, such as wikis, Google Apps, Google Groups had been used for educational purposes, and some participants were thus already familiar with their educational uses, in the same way as they were familiar with the platforms that Sakai was using for its own communication purposes. Overall, Sakai 3 was pursuing the adaptation of a broad set of general purpose web 2.0 offerings to the educational context, and the focus of the review was guided by this overarching goal.

Contextual scenarios

The scenario collection was initiated by Daphne Ogle on the basis of user-centered methodologies, with the goal of gaining a better understanding of groups in the new Sakai:

> "Our goal here is to capture a diverse set that will help us understand group functions across Sakai."[151]

While the collaborative spaces of the Sakai community do not indicate any direct uses of these scenarios, they clearly shared in the overarching motivation of all other collections undertaken in the same period to gain insight into the contexts of use for the purpose of sense-making and conceptual construction. Besides Daphne Ogle, three other participants contributed to the collection.

Most importantly, scenarios in software design were generally formulated to illustrate how a software system would be used to accomplish goals beyond the immediate interaction with the interface. Scenario writing was explicitly invoking participants' imagination for formulating short stories around educational and research-oriented use of Sakai. At the same time, it also presupposed that imagination would be conceptually grounded in several ways. On the one hand, scenarios were meant to be grounded in realistic contextual detail. They did not have to be based on real events, but they had to be such that they could have happened. On the other hand, scenarios were also conceptually grounded in, as they were expected to be motivated by noteworthy patterns.

Consider the following story:

> "Robin, a professor in the Business School, encourages her students to do a significant amount of group work in her International Business course. Several problems during a term are introduced to students who then may work in teams of 3-5 and need to pull together information from a variety of resources in order to fully address the problem. Student presentations of their work on the problems may be included. Since they often won't get to choose their teammates when they are out in the work world, Robin likes to coordinate the teams herself paying

some attention to class schedules, and where students live. Later in the term, a couple problems require each team to have at least one person who has already taken Business Law so she is more concerned with that criteria than the convenience factor when she creates those teams." [152]

This story may be said to be typical and contextually truthful because we know that institutions of higher education commonly have schools of business, with professors, students, classes, semesters, assignments. It makes sense when we hear that there is a class in Business Law and the professor organizes project work. As a matter of fact, it may also be suggested that team-based project work is a method of preference in business schools. Business students are often, mature, working adults, who need to juggle class schedules with other obligations in life, so it is reasonable that the professor is attentive to these details. Then again, it is also reasonable that other considerations, such as existing skills, may also be taken into account. For all of these details, the story was relying on knowledge about the world of education.

On the other hand, the story was written to convey a point about the challenges specific to the Sakai platform: it is a reasonable requirement for the online system to allow for the ad hoc grouping of students within a class for the purpose of team-based collaboration, and it is useful if the system can support the process by making available criteria for grouping. It was aligned with an underlying conceptual model of the interface, which served as an implicit grounding for putting the narrative together. Note that the model itself was not made explicit, and it was never spelled out in terms of concrete details of the interface. It was operational at the level of conceptual abstraction: means of grouping are needed alongside criteria to support this activity.

Most of the scenarios were fictitious stories guided by conceptual understandings of the world of the education and the interface. Others entertained even stronger links with reality, either because they were grounded in a occurrence, or because they were envisioned to become a reality sooner or later. The following narrative is based on an account recounted by an instructor:

> "I divided students into 38 groups of about 5; they put in a request to me about which students they wanted to be grouped with, but they also had the option of having me group them by default if they didn't put in a request. I then wanted to assign them to individual groups, and the groups function seemed great to me. But I had some problems/confusion:
>
> […]
>
> * Second, I royally screwed up the communication in several ways. I set up an announcement for each individual group. I had copied their names and e-mail addresses from the group roster and put them in each announcement text (because I couldn't understand from the "help" and "how to guides" how and if the students would/could communicate among themselves via messages on bspace [the University of Berkeley's Sakai portal]; I copied names and e-mail addresses into the announcement because I was worried otherwise they wouldn't be able to get in touch with each other – is there some additional help that could be provided that could answer this question on the website?). In addition, I selected on the announcement the e-mail notification as high, to all participants. I assumed that this meant *all participants in the group*! But it went out to all students – all 38 of my individual group announcements went out to the whole class. Yikes, I

> really misunderstood that. […] Not only was this pretty embarrassing but I am worried about privacy issues."

The story described what the user was attempting to do, how it could be achieved by the means that the platform made available, and what troubles a poorly designed interface was causing. The situation described was similar to the fictitious scenario presented before, with the instructor creating student teams within a class. This time, the educational context was real, and the narrator was making a point about the underlying structure of the interface, more specifically how it was not aligned with the structure she expected to be there in light of her task.

Other stories represented what may be called willed fiction: more or less concrete plans that an institution was already pursuing at the time of the writing. On this particular page, these scenarios were all described by NYU, the institution which eventually took the lead in finalizing and piloting Sakai 3 in the managed project.

> "NYU will create shared collaborative spaces for each of its global study abroad sites [actual physical sites]. These [spaces] will be used to provide students access to information relevant to their specific global site, such as maps, visa information. Students will use it as their "community" space to self-organize shared events and find out about formally hosted events for their site, share location information such as a great restaurant; etc. Students actively enrolled in a study abroad location would be automatically enrolled in this common site. Newly admitted students could join to learn about the experiences abroad."

The content of this story was similar to the other two examples in that it was combining details of the educational context with glimpses of an interface inserted amongst the

activities of this context. There was mention of a university, with a study abroad program, students, enrollment, admission, institutional events from the educational domain, and students were described as accessing information in a site, self-organizing, sharing location information in a foreign setting, being notified about events, as well as sharing and learning about new experiences in the program.

Willed fiction accounts were situated mid-way between true stories and fictitious scenarios. They took their grounding from the detail of the educational context, just like fictive scenarios, but in their motivation they could claim to be closer to true, since they were willed by actual people in actual situations. These people were not necessarily the ones that the scenarios would serve: in the case above, the scenario was about helping students, but it was not formulated by students. Given the large-scale, role-governed, hierarchical character of education, being part of an institutional context for real is probably the most any speaker would be able to claim. At the same time, the envisioned situations had not gone through the test of realization.

Overall, the scenarios outlined in this collection represented constructed accounts of meaningful situations, which drew their conceptual motivation from two domains: from the educational context on the one hand, and from interactions with an interface on the other. They were weaving together meaningful bits and pieces from these domains in such a manner as to make a point about the educational use of an interface. For doing this, they were relying on conceptual understandings about the world of education to project a conceptual model of the interface. This model remained on an abstract level, without providing concrete practical details. Understandings of the educational context were more or less removed from the world of education: some narratives were grounded

in actual and concrete past occurrences, some in actual situations of envisioning, while all of them were drawing on understandings gained through sustained experiences, or what may be shortly termed as knowledge. Table 8.4 provides a summary of the various types epistemic grounding that were present in the narratives.

Table 8.4: Epistemic grounding of contextual scenarios in terms of educational context and software

Types of narrative	Educational context	Software
Account of past occurrence	Experience of concrete events	Concrete interaction
Willful fiction	Sustained experience with the world of education	Conceptual model of a possible interface
Fictive scenario		

8.3.5. Discussion of the first case study

I suggest that we pull together the threads of analysis drawn from the various collection efforts in the Groups Project. My claim is that in the conceptual design space unfolding from the Sakai 3 vision, participants were routinely navigating across the different shades of imagination and real-world understandings as they were pulling in conceptual understandings at various orders of distance from concrete experiences to feed the effort of sense-making and conceptual modeling. Thus, on the one hand I have shown that understandings about the contexts of use were visited in connection with conceptual construction, to broaden the pool of understandings within which sense-making could operate. The Sakai 3 vision outlined an interest in educational and socio-technical contexts. Conceptual construction was unfolding at the confluence of these two domains. The exploration of the socio-technical domain was typically focused on activities with the

user interface as a means of achieving user goals. On the other hand, I have shown that the ensuing conceptual collections were spanning a broad range of forms both in terms of conceptual abstraction and grounding in experiences. Explorations of conceptual models coexisted with concrete, practical details, specific events with typical occurrences, and first-hand personal experience with the accounts of others. The particulars often came embedded in conceptual models, and they could be omitted in processes of reasoning, as the various conceptual collections suggested. Participants were routinely switching and thinking across the various forms of understandings.

It also appears that underlying the engagement with the various conceptual forms was a layer of general knowledge about the world of education and software. While this knowledge came alive in individual thought processes, it was commonly seen as generally shared and sharable, providing a general foundation for collaborative conceptual efforts. Participants were interviewed, contributions shared, reworked, and unclear conceptual points discussed by bringing in further understandings. It was also commonly assumed that knowledge of higher education has currency beyond the immediate contexts of collaboration, within the Sakai community and in the world of higher education. While collections efforts within the Groups project did not venture beyond the close circle of Sakai contributors, other collections, such as the Investigation project described previously, were seeking to tap into the understandings of instructors. Many Sakai participants were also routinely hearing about first hand experiences of members of higher education as instructional software technologists at these institutions.

I have also shown that the various forms of understandings were drawn into the design space to support the sense-making and conceptual construction around Sakai 3.

The efforts of collection and construction were separate, but connected. Collection was a sustained activity which was planned and organized on its own right, but clearly with the purpose of helping to make sense of groups for Sakai 3. It was relying on conceptual models, and it was also bringing into motion creative modeling, but it was in and by itself not so much oriented towards building new models as bringing into play established ones. Participants were able to switch between these modes of thought, but did not pursue them at the same time; instead, they were reflectively aligned in terms of a shared design orientation, and the related demands of sense-making.

Related to the alignment with design, the perspective of the user interface played a significant role in guiding and framing the content of collections. Contextual scenarios, benchmarking of existing interfaces, federated authorization and mental model collections all included accounts of activities around the user interface. The framing followed from the UX-driven prototyping process, which was itself rooted in user-centered methodologies.

8.4. Second case study: Memory practices and the contexts of use

The first case study about the Groups Project was focusing on a series of collections which were directed at pulling understandings about the context of use into the design space. While these collections were guided by the need to make sense of the design space, they were also pursued in anticipation of future uses of the collected materials to inform design. Contributors were drawing on various sources of understandings about the contexts of use, including their own knowledge and memory, to create a textual storage of relevant useful understandings related to groups. This knowledge base served as a centralized location of understandings that have been previously scattered across the

community, in participants' heads and in the various platforms of communication, while also providing a framing for their formulation.

To show how Sakai was strategically drawing on and recreating its knowledge bases about the contexts of use, I will now revisit a collection initiative within the Content Authoring project[153], which had been cursorily described in connection with conceptual modeling. The project itself took place in 2008, when the Sakai 3 vision was taking shape, and plans for a new artifact were in the making. The focus of the collection was content, and it was undertaken in preparation for the Content Authoring Summit, to survey uses of structured content in higher education. Over the period of less than a month, the following imaginary scenarios were added to the collection:

- a professor creating structured content for a course;
- an instructor making a syllabus with the OSYL (Open Syllabus) tool in Sakai 2;
- a professor reusing structured content from external repositories, relying on the SCORM and OCW standards;
- a professor using the SOUSA tool developed for Sakai 2 to create a structured presentation from his course materials;
- a librarian making a Research Guide with the help of the Sakaibrary tool created for Sakai 2;
- a student making his portfolio with the help of the Portfolio tool created for Sakai 2;
- an instructor and a student making a MyProfile page for Sakai 2;
- an instructor using the WebDAV file upload technology within Sakai2.

The Content Authoring project was initiated to bring together Sakai members who had shown an interest in structured content, and may have also engaged in developing related tools for Sakai 2. Participants were aware of the different initiatives and the persons involved, so this was a collection by invitation, where contributors were describing their own areas of expertise.

In the different original projects, levels and modes of engagement were varied; Open Syllabus, Portfolio and Sakaibrary were large, highly visible projects, which were embedded in the relevant communities of practice, and known for keeping up a lively conversation with the contexts of use. Sousa was a one-person project created by a developer not embedded within higher education, and thus characterized by a technical edge. The SCORM and WebDAV stories were based on software tools, coming similarly from a technical orientation. Overall, the collection brought together heterogeneous perspectives on the common platform of imagined scenarios. At the same time, it also created a springboard for reaching out to the local knowledge bases of the projects, which were also linked to the Content Authoring home page.

In light of the above, the purpose of the collection was to bring together the practical knowledge of Sakai participants on structured content, and make it available as a shared resource for the purpose of collaborative thinking at the Content Authoring Summit and beyond. In particular, the ideas discussed at the summit were fuelling a series of spontaneous conceptual modeling efforts around structured content, which have been described in Chapter 3. The discussions were eventually culminating in Korcuska's Sakai 3 proposal, and its strong focus on content, which was later inherited by the 3akai project.

In the Content Authoring project, collection was used as a strategy for tapping into the expertise of the community for surveying what was working in the contexts of use as well as what was possible in terms of technology. The rationale was to reuse the expertise distributed in the Sakai community as a springboard for outlining a novel system. It was based on the idea that existing projects represented a local concentration of knowledge about their areas of interest, stemming from direct engagement with the relevant educational and technical contexts. The knowledge base of these projects could be reused, both directly, by involving knowledgeable participants, and indirectly, by engaging what has been discussed and produced within the projects. The overall goal was to mediate the contexts by means of the projects that have conceptually processed and reworked them, and taking their accomplishments to a new level within the Sakai 3 design space.

Turning the Sakai community into a knowledge base for the purpose of reaching back and planning forward was a large-scale strategy which could be seen at work in various other efforts related to Sakai 3. I will now broaden the perspective of my analysis, and look at Sakai's strategic efforts at the community level for making knowledge about the contexts of use available for the design of software. These efforts gained their power the social distribution of knowing among participants. I will suggest that the socially distributed approach to knowing the contexts of use became especially powerful when it was coupled with the cognitive processes I have described in the first case study of the chapter.

In chapter 4, I have described Sakai as a special type of open source community which was conceived on two levels. On an abstract institutional level, it was a

collaboration of institutions of higher education. On a practical level, it was a community of participants delegated by the institutions. Sakai was conceived to be primarily a community of educational technologists employed in academia. Participants were not end-users themselves, but they were in positions within their respective organizations where they routinely engaged with the contexts of use in a range of qualities. They routinely met, observed, and communicated with instructors. They also routinely faced problem-solving situations where users needed support in very real situations, and became informed about the details of these situations. Overall, they were in strategic positions which gave them access to a wide pool of understandings about the contexts of use from within the organizations.

Participants in turn created opportunities for engaging local users in a focused and systematic manner. A major strategy involved learning from structured inputs from end users, through organized feedback or research based on interviews, questionnaires, error report forms or similar tools. This mode of learning was rooted in the ethnographic methods of fieldwork. The Investigation project described in Chapter 6 was a major research endeavor along these lines. The research was designed following the principles that the Goal-Directed approach outlined for fieldwork, with strong foundations in quantitative research methods in the social sciences. Sakai participants conducted interviews with instructors, students and persons in other instructional roles at their own institution. The general idea was to obtain rich information about the context of web-based assignments, and the related goals and activities of participants in various roles. The same interview guide was used as the template for these interviews across all participating universities. This large scale research involved considerable organization,

for which the Sakai community provided the necessary background. The idea of using ethnographic methods came from one of the participating institutions. Flow Interactive was a design community at Berkeley, and they had been actively sharing the outputs from a similar research effort conducted in the early days of the Sakai community.

Sakai 3 incorporated an attempt at a radically different approach for engaging end-users, which would allow involving instructors and students as developers and designers of software add-ons for the platform. The modular architecture of Sakai 2's platform was designed to make possible small-scale contribution in the form of tools. Sakai 3 was nurturing a parallel, but more radical idea of modular contribution in connection with the widget-based architecture. This approach opened up the possibility of contribution on a much smaller scale, in the form of small segments of functionality within widgets. The idea of opening up the platform to contribution from regular end-users was present from the early days of the Sakai 3 vision, and as the standard was taking shape, it was implemented by the front-end team from their own initiative. In this manner, the design of widget-based modularity was never the focus of community-wide discussions. It came to the fore after the first release of S/OAE in 2012. The collapse of the managed project meant an opportunity for reinventing the priorities of the new platform, and the possibility of grounds-up contribution gained central place in this process. Widget-based contribution has remained a significant promise of what is now Apereo OAE to directly engage end-users in the making of software. The hope has been that the system would be able to support a truly user-friendly process of software development, which would allow end-users to create the software functionality they needed for their own specific ways of working. The project was also planning to make

available a widget library, with the possibility of finding useful widgets, which could be installed and even modified by end-users. In this model, knowledge of the contexts of use was conceived to be channeled directly to the development of useful software functionality. The mediating role of Sakai participants would be exercised through the modular platform that they had designed and developed.

The community that had flourished under the Sakai Foundation made available further avenues of learning about the contexts of use. First of all, participants constantly organized opportunities for *sharing* knowledge, and learning from experts like themselves. These included the yearly Sakai conference, workshops organized on a smaller scale, like the Content Authoring Summit, and the various local initiatives of collection I have pointed out in the case studies. Email and other communication channels were also used for polling the community in an ad hoc manner. Sharing knowledge in this manner was based on the idea that participants were important sources of first-hand knowledge and conceptual understandings stored in their personal memory about the contexts of use.

The various collection initiatives, like the collection of content authoring approaches in Sakai described above, the collection of widget-based scenarios (described in Chapter 5), or the collection of scenarios under the Minispec effort (mentioned in Chapter 6), placed the sharing of experiences within a broader perspective, which connected them with the momentarily relevant challenges of design. Conceptual understandings about the contexts of use were formulated in accordance with the design-related framing, as use case or scenario narratives, or even as a metaphorical account of mental models. In the first case study, I have described this as a process where personal

understandings of the contexts of use were visited in connection with conceptual construction. I will now add to this analysis that Sakai as a community for collaboration created the social context within which these cognitive processes became possible. In other words, the cognitive work of bringing conceptual models of context to bear on design was tightly coupled with social distribution. In light of this, collection may be understood as the local epistemic strategy which created avenues of conceptual construction in terms of a sharing of personal understandings.

It may be argued that participatory design routinely adopts a similar epistemic strategy, which channels the knowledge of potential or actual end users by engaging them in the cognitive processes of conceptual construction underpinning design. The strategy of Sakai as an established community went beyond the approach of participatory design in one important respect: it gave an extended temporal dimension to local collaboration. Temporal distribution was relying on the use of communication platforms as the means of creating an external memory. Local initiatives and spontaneous discussions were tapping into the pool of knowledge that the community had created. Collection efforts in particular revisited knowledge bases that had been created earlier: the Content Authoring collection engaged with the knowledge base of related projects, the Widget-scenarios and the Minispecs pulled in the personas created for unrelated purposes. Collection usually also had a forward-looking framing in the sense that digital collections were created with an eye to subsequent activities that will be able to rely on them. Thus, establishment of a community around Sakai created a long-term horizon for collaboration and sharing, which made available temporally distributed strategies for learning about the contexts of use.

Finally, and perhaps less obviously, connecting the community of Sakai with a domain of experience made way for an epistemic strategy that decoupled knowledge about the contexts of use from the direct, conceptually unmediated experience of an individual knower, and connected it to shared conceptual understandings of the domain. Participants were entering discussions with the assumption that despite differences in the details of local implementation, higher education was a domain characterized by a pool of shared knowledge, which made mutual understanding and collaborative sense-making possible. Knowledge did not reside with a singular group of knowers, and it was not captured in a privileged single expression, it was instead located in the unceasing sense-making performed within the community, wherein personal conceptual models of higher education were shared and became continually reworked in light of conceptual understandings of others.

To illustrate this point further, I will quickly revisit the exchange described in Chapter 8, where the lead of the back-end was complaining about the lack of a single definition of group and space, which could express the requirements of the community. His initial request was met by a series of explanations detailing the conceptual challenges of making sense of groups in the higher education context, and a historical account of how these conceptual challenges were collaboratively sorted out. At that point, the community had come to a common conceptual understanding of groups, which found significant expressions in written and graphical accounts of the group dashboard. At the same time, individual understandings had their own perspectival nuances. Because of this, responders found it important to make private conceptual ‘footnotes’ to their

explanations, an approach which exasperated the back-end lead, who wanted to hear a straightforward definition.

Overall, the community of Sakai followed a distributed cognitive process of design, in which conceptual construction was distributed in time and among participants, and it was acknowledged to have foundations in shared conceptual understandings of the domain of education. The various forms of distribution reflect the strategic decision of interpreting participation in terms of an institutional domain, and the community-based distributed development model of open source. The community of Sakai served as a broader organizational framing in which epistemic strategies of distributing the work of design became possible. In my analysis of the various episodes, I have shown that these strategies were conceived and orchestrated in the local contexts by participants; the distributed approach should thus be seen as an epistemic strategy that designers can routinely rely on to create the possibility of their own work of construction.

8.5. Discussion: reconfiguring the empirical and analytic stance in the social approach to software design

Throughout the chapters, I have provided an account of software design as a sense-making process rooted in human cognition. In the above, I have argued that this sense-making relies on the understandings that people have about the contexts of use, and broadening the pool of understandings that is available for the sense-making process is a major epistemic strategy for furthering the work of design. This account ties the acquisition of knowledge about the world to the conceptual processes involved in the design of software. Knowing the contexts of use is important, but the understandings that

become productive are those that are implicated in the conceptual processes of design, which also means that new knowledge will be pulled in in accordance with the unfolding cognitive process. Related to this, it may be emphasized that the knowledge gained directly from experience is itself tied to conceptual organization, and it becomes implicated in sense-making and conceptual construction on account of these conceptual foundations.

I have also shown that going out directly to the world for further insight is only one of the ways in which relevant understandings may be pulled into design. Within Sakai, understandings could be indirectly gained from others with experience in the field of education, as well as by revisiting personal or collective experiences in a systematic manner for making scattered knowledge available locally was equally important. The DCog outlook suggests that knowledge can be mediated by means of a combination of temporal strategies of archiving (in the form of personal and external memory), and social strategies of sharing.

The above account suggests that the emphasis that social accounts have placed on the empirical nature of their contribution is misguided in two important respects. First of all, knowledge appears to be tied in to the generative thinking processes of designing, which suggests that an empirical strategy for software design should start from an understanding of the underlying conceptual process. Second, knowledge can be pulled into design indirectly, which suggests that the related strategy should rely on the opportunities of mediation that a DCog outlook offers. In this respect, Sakai's strategy of creating a community of participants who were potentially both 'knowers' and 'designers' to some extent appears promising. This suggests that social researchers need

to become designers to some extent to bring their insight into software design, or reversely, developers of software need to become socially curious, to bring social insight to their work. The existence of a platform of sharing, such as a discipline, a profession or an open-source community makes possible a division of labor where everybody partakes of the conceptual foundations that make design possible, but there are differences in the extent to which participants are involved in the various activities of 'knowing' and 'design'.

Going further, the analysis above also suggests that human-centered attempts at making role for analytic thinking have also missed their point. Social approaches should not seek to describe full-fledged implications for design, but they should not continue doing social research in the traditional sense, either. In Chapter 7, I pointed to an apparent lack of social conceptualizations in software design, and gave a tentative outline for what social-technical imagination may be like with respect to social software. My suggestion is that the epistemic strategy of social approaches should start from a social-technical imagination, and approach the problem of knowledge about the contexts of use in close connection with the epistemic stance of conceptual construction. This implies a wholesale reconfiguration of empirical and analytic forms of thinking and a shift from theoretical to design-oriented conceptualizations.

I suggest that an account of social-technical software design should start from the notion that a software system is being designed from a social perspective to the extent that it is being imagined in a tentative manner in terms of the patternings of the human activities that implicate the system. Thus, the social-technical imagination differs fundamentally from social theory at large in that it is the conceptualization of a future

system. At the same time, it also relies on the general epistemic stance of social imagination, which connects individual action with broader patternings and interdependencies in human activity. In my sketch of the social-technical imagination I have focused on the role of framing devices, whose role is to connect imaginative thinking about social patternings of human action with the evolving artifact. I will now take the exploration of social conceptualizations involving artifacts further by suggesting that these forms of thought can be imagined as a form of heterogeneous engineering, along the lines that the social theory actor-networks has defined the very notion of the social phenomenon.

In a theoretically oriented summary of the positions of actor-network theory, John Law (1992) formulated the following definition of the social:

> "the social is nothing other than patterned networks of heterogeneous materials. This is a radical claim because it says that these networks are composed not only of people, but also of machines, animals, texts, money, architectures -- any material that you care to mention. So the argument is that the stuff of the social isn't simply human." (p. 380)

Actor-networks have indeed been described as concatenations of human and non-human agents, which together have the effects that have been traditionally perceived as the social phenomenon, the purview of sociology. Law continues with a description of the task of sociology:

> "So in this view the task of sociology is to characterise these networks in their heterogeneity, and explore how it is that they come to be patterned to generate effects like organisations, inequality and power." (p. 381)

In other words, the task of sociology is to conceptualize patternings that implicate the entanglement of humans and artifacts. Taking this characterization further, I suggest that the task of social-technical design may be understood as the imagination of possible new ways of patterning (or de-patterning, for that matter) in the heterogeneous actor-networks of humans and artifacts. This is similar to what John Law described elsewhere as heterogeneous engineering (1987), or the production of new social phenomena by means of relying on artifacts. The significant distinction is that it is an imaginative mode of production, which does not rely on knowledge of the world with the goal to create analytic accounts, but uses this knowledge in constructing and exploring dynamic thought experiments about possible, but yet non-existent forms of software-based realities.

CHAPTER 9. INFRASTRUCTURAL IMPLOSION

9.1. Designing within an evolving software landscape

Human-centered approaches have been increasingly aware of working on shifting ground. This awareness has been expressed in terms of the distinct epistemic perspectives of consecutive waves in HCI, notably the 'Third wave' (Bødker, 2006; Harrison, Tatar, Sengers, 2007) and the Ubicomb vision (Dourish & Bell, 2012; Abowd & Mynatt, 2000), and most recently in speculations about what an imminent fourth wave may look like (Abowd, 2014; Dourish & Bell, 2012). The shifting context is conventionally indexed to Moore's law (Grudin, 2012), which describes an exponential growth rate in the computing power of chips. This growth also impacts the size of computing hardware, and has made possible the proliferation and diversification of computationally powerful devices at the human scale. The expansion of connectivity, most notably over the web, has been another significant source of change (Dourish & Bell, 2012). Thus, the computing context of Ubicomp or the Third wave in HCI have been described in terms of more people owning more and diverse devices, with mobile and ubiquitous access to the network, which together result in an expanding ecosystem of devices and supporting infrastructures (Edwards & Grinter, 2001; Grinter, Edwards, Newman, Ducheneaut, 2005). Social ubiquity has brought a general cultural embedding of computing, which has been translated in Ubicomp and the Third wave as the cultural diversification of design, with an increased attention to meaning and the phenomenology of experience (Dourish & Bell, 2012; Harrison et al., 2007; Bødker, 2006).

The various human-centered reflections also suggest that being conditioned by opportunity has become an identity based on formative experiences within the field.

There is a sense in the community that the field is by necessity reactive to its broader computing environment. Vision has emerged as a tool for engaging with the challenge of shifting foundations in the future-oriented context of design. Vision has come to be perceived as the means to provide direction and purposiveness to human-centered approaches, and to create the cohesiveness of the field. Dourish and Bell (2007; 2008; 2011) have written extensively about the role of Weiser's vision for the Ubicomp field, and advocated for the reimagining of the vision ("divining a digital future") to move the research and design project of HCI further. Similarly, Abowd (2012; 2014) has suggested that the maturity and generalization of ubiquitous computing both as design practice and experience calls for the formulation of novel visions on what the fourth wave of computing may be like.

At the same time, little attention has been paid to the details of the conditioning of human-centered approaches, notably in terms of the configuration of human-centered practices within the broader landscape of computing. Authors such as Dourish (2006) or Suchman (2002) have written emphatically about the local configuration of ethnography and social research within the organizational context of design, but the epistemic consequences of its placement within a broader socio-technical context have remained unexplored. This chapter will explore the role that an evolving context of open-source software was playing in the design of Sakai 3. I will suggest that software infrastructure played a significant role in mediating contextual change to the project of the new Sakai. The case study will show how the adoption of back-end components contributed to the formulation of the Sakai 3 vision and the formation of the design space.

In Chapter 8 I have told the development story of Sakai 3's kernel. I will now revisit this story from the perspective of the broader context of networked software, and the open source focus on sharing infrastructural software components. The story of the kernel has shown that the development of the back-end reached out extensively to available open source components, which were identified and validated in a process mixing collaborative reasoning with tinkering. I will start by giving a more detailed account of these practices in terms of an epistemic strategy, and suggest that they indicate the existence of a culture of modularity in open-source software development, with a division of labor where back-end technologies are expected to translate the evolution of computing to generic patterns of use. In this manner, component technologies mediate an evolving software context to the local setting of design. In the second part of the case study, I will show how component technologies contributed to the formulation of the design space around Sakai 3 by creating plausible promise for the elaboration of their generic template of use. I will introduce the term implosive infrastructure to refer to this contribution.

In the discussion, I will situate implosive infrastructure amongst evolutionary models in technology described by SCOT. I will argue that these models reference the social patternings that result from the different types of technologies, and suggest that the modular design of Sakai 3 results in a characteristic combination of patternings which is constitutive of what the new artifact is. In other words, I will argue that Sakai 3 is not easily accounted for in terms of the kinds of uses that it offers, and to make sense of what it does as a socio-technical artifact, it is best approached from the perspective of its making, which involved the layering of different approaches and perspectives on design.

9.2. Case study

9.2.1. Practical reasons for the use of open-source components

Adoption of standards-based open-source components was a familiar strategy with positive connotations for server-side developers within the Sakai community. This was clear from a recurrent set of reasons commonly cited in connection with the adoption of components, and the interested awareness that server-side developers exhibited for them. The various reasons could be cited in combination, or pitted against each other in heated discussions. The reasons were the following: (1) standard interfaces allow for choice in (a) implementation and (b) complimentary technologies, and give latitude for (2) the varied local ecosystems of technologies at the institutions adopting Sakai 3, and (3) the possibility of making better choices over time; (4) open source components enjoy wide adoption, which means that (a) they are production-tested, (b) errors are spotted and corrected by the community, and (c) the software becomes improved; (5) open source components rely on an advanced level of industry knowledge and experience, (6) which the Sakai community would not be able to replicate on its own; this advanced knowledge appears in (7) a deep understanding of requirements and (8) in quality code which is (a) simple, and thus appropriate for developer collaboration; (9) open source components save Sakai 3 developers from a replication of efforts in areas that enjoy broader attention within the software development community, and free them for focusing on the implementation of specific, local features.

Many of these reasons appeared in John Norman's document about the rationale for adopting a JSR-170 compatible implementation for content management. The argument was unfolded from the following two questions:

"Why is a JCR-backed Content Service a good idea?

We have a stable service for content already (CHS: Content Hosting Service) - Why change?"[154]

The following answers were given in the document – I will list the number of corresponding reasons described previously:

(**8a**) "1. CHS is now a large, complex and difficult-to-maintain block of code. […]

(**1a**) 2. Implementing a Java standard interface should allow us to substitute various commercial and open source implementations of content repository, giving us

(**3**) • some degree of 'future proofing' so if one implementation (say Jackrabbit) becomes noticeably faster than another implementation (say Xythos), institutions can switch without major disruption

(**2**) • different institutions with different budgets can make different choices […] among alternative repositories without significantly affecting the tools that use content hosting

(**1b**) • a richer set of opportunities for other campus integrations that can work against the same store or provide different views of it

(**7**) 3. JCR repositories already implement some advanced features that we want (e.g. versioning) and using these implementations […] will save considerable

(**9**) development effort and reduce deployment risk (because large parts of the code

(**4a**) are already production tested).

(**5, 6**) 4. The variety of JCR implementations available all exceed what we have been

(**4a**) able to achieve in terms of quality, test coverage, performance, etc.

(**7**) 5. JCR API has emerged from a great deal of industry knowledge exchange on the requirements of a content repository interface and therefore represents a mature,

(**8b**) relatively complete service, that is noticeably simpler than the current Sakai service.

(**1b**) 6. Whole new blocks of functionality may become easily available to Sakai if they are built on top of JCR (e.g. Alfresco Document Management)." [155]

Local development effort was a major concern: participants had concerns about the availability of local development expertise and the amount of developer time available. Selling the rewrite of Sakai 2's kernel was thus based on the notion that open-source could do better than the existing home-grown solution, as John Norman suggested, or a local rewrite, as Ian Boston argued:

> John Norman:
>
> "As I understand it, there has been a feeling for some time we should not be seeking to maintain our own component manager when there are good solutions available from other sources. I'm not sure we should be investing our scarce resources in continuing to develop a custom Sakai component manager if it is better solved elsewhere by others."[156]
>
> Ian Boston:
>
> "If we believe that a large part of Sakai is essentially a[n] X with customizations for education and research, and we can find a component that delivers this functionality and passes our criteria, then we spend time integrating that component rather than writing our own." [157]

Similarly, Korcuska reasoned that in case no open source candidate component was available, an open but custom solution was still better than writing code locally:

> "Assuming there isn't a good candidate in the "open but custom" category then I'm not worried about the relative lack of JCR implementations. Because the only other choice is to do it ourselves. / And I don't believe will do it better in the time allotted or in the long run. "[158]

What is more, the open source ecosystem presented itself as an approach where the adoption of one component should allow saving development with complimentary modules:

> "Having bound to JPA we can use Cayenne3, OpenJPA, Hibernate, JPOX, and others, by changing about 20 lines of code. At the moment EclipseLink *looks* like a front runner in terms of performance, features and usability.
>
> Having bound to Spring […] we can use ..... Spring, that's what I mean by proprietary standard."[159]

Thus, openness of open-source not only meant a saving of effort, it also allowed tapping into a broader pool of software.

9.2.2. Epistemic engagement with open source

Reasons for open source were formulated as strong expectations, which had to be validated against the local context. This is why arguments often used the modality of 'should', as we have seen in John Norman's document, or in Korcuska's reasoning above. In Chapter 7 I have also shown how decisions about the adoption of a new component were preceded by a local implementation, which necessarily also involved hands-on engagement with the code. First and foremost, as the following quote by a

back-end developer suggests, engagement with the code represented a basic, unquestioned strategy of epistemic mastery for developers:

> "The[n] I started to evaluate spring/spring-dm and their solution just kinda made sense after working with it for a little bit." [160]

The basic practice of tinkering was further translated into a reasoned epistemic strategy, where implementation meant evaluation by proof of concept. The majority of proof-of-concept prototyping efforts were spearheaded by Ian Boston, who argued vocally that "Real evidence means working code":

> "I agree that we should base the analysis on evidence, and we should leave no stone unturned in gathering that evidence. Which means that we should write code that discovers exactly what OSGi will look like and how it will work, rather than just talking about it.
>
> This means writing java code that explores [the] issues, analyzing the results rather than just saying that X is the solution. […]
>
> But, the work I have been doing, is just one possible implementation and I am doing it because I don't want this group to argue itself into any single solution without some real evidence that that solution will work. Real evidence means working code. With working code that we have all had a chance to review and explore, possibly with some of the users external to this group (UI Developers, Production) then we can make an informed decision that will standup to the community."[161]

Ian Boston emphasized achieving the evidence of working code, making a component work within Sakai 2's existing ecosystem of code. Implementation, as John Norman

suggested, was also a means of testing expectations that the open source code was doing what it promised to do and it was doing this well:

> "[…] I support Ian's approach - try things out and make decisions based on evidence. We (as a community) have been caught out before by the allure of technology that we are not familiar with, but which is the 'thing to do' of the moment. I am concerned that OSGi as a whole may be a similar mirage and am encouraging Ian in taking a sceptical approach.
>
> […] Our objective is NOT to use OSGi at any price, it is to have a performant, safe, managable component manager which preferably is maintained by a large group of deploying 3rd party organisations."[162]

While code quality was a strong expectation with respect to open source, both of the above arguments suggested that it had to be validated in the local context.

Implementation of others' code could bring challenges of its own. Code complexity was an important aspect of this, and as I have shown in Chapter 7, the difficulty that Sakai developers may encounter on a daily basis in working with components was an important consideration. These issues became acute in connection with the adoption of an initial set of components. One of these was OSGi, a module for launching other components and managing their runtime interdependencies. OSGi underwent an initial evaluation, which turned out to be longer and more painful than expected. A number of back-end developers, who were invited to work on the local implementation, admitted that they did not have enough familiarity with the domain of web containers to directly engage with the implementation of OSGi:

> "As one of those toe dippers, I'm 'biding my time' waiting for something to which I can usefully contribute. The existing conversation is at a level of technical detail at which I don't feel qualified to comment."[163]
>
> "That's my fault and I am not anywhere close to report my findings in a professional manner as Ian did. I think one of my issues is that I don't have an in depth understanding of web containers, and anything else for that matter :) , as Ian does.
>
> […]
>
> I think one issue is that as I pointed out above, I don't quite understand your solution [OSGi] and am concerned about how much I, I don't want to speak for others, will be able to contribute if we choose one solution over the other. Maybe this will change once I, we, have a better understanding/appreciation of your solution. I guess the key is as you pointed out, that it should be as easy as possible to write legacy and new style services and tools."[164]
>
> "Thomas and John have mentioned the complexity of the OSGi-inside (cue Intel jingle) approach.... I don't think that is evidence that is the wrong direction .... we have a complex problem to solve […] if we want to remove our dependency on others, we'll have to blaze a little trail and hopefully we could learn enough to gain the head room to make an elegant simple solution for developers"[165]

The comments sounded an optimistic note in referring to the possibility of learning; they cited practical engagement with the code and the experience of trailblazers within Sakai as possible sources of learning. Engaging with OSGi actually resulted in practical

knowledge which made way for a partial local implementation relying on OSGi's implementation strategies:

> "In the mean time we have a simplified component manager that borrows some concepts from OSGi, like a classloader search policy, activation and exporting of packages from components. The approach does not create tensions between the component manager and the webapp container in the same way that OSGi does leaving the UI developers oblivious to the change in underlying kernel."[166]

Besides practical engagement with the code, monitoring the open-source scene was also a major source of understandings. I will now describe how this practice involved a broader epistemic engagement with implementation strategies that went beyond the code.

Keeping an eye on the open source community was an activity that many back-end developers could be observed to be involved in. On a practical level, monitoring the open-source scene involved a variety of interests, ranging from reports of new developments and production implementations to discussions on the robustness or the strategic direction of development communities. I will have to add that there was no collaborative forum for sharing or discussing these issues within Sakai. This means that the pursuits mostly remained private, but they could be inferred from focused discussions, such as those on the choice of components for the new kernel. At one point during this discussion, Thomas Amsler gave the following account of how he went about "researching what's out there and where the industry is heading"[167]:

> Exploring the feature landscape of components:
>
> "I am worried about standards too, but it looks like that a lot of the spring-dm solutions will end up in the upcoming (R4.2/R5) OSGI specifications."

Looking at the components that important players use:

"I also found some information that BEA and Oracle and SpringSource itself use spring-dm for some of their products."

Looking at implementation examples:

"found it hard to work with because of lack of examples and not knowing if anybody in the industry is using this approach, at least I haven't come across any information with respect to that."

The elaborate arguments that unfolded around the suitability of components also suggest that keeping up-to-date with the open source scene required the construction of an opinion from partial information and secondary sources. As Clay Fenlason suggested, this involved:

"taking secondhand reports of the state of the industry and other projects as a proxy for an informed opinion. Perhaps necessarily so: we aren't, after all, purely a research collaboration, and a consensus of other, respectable projects is a valuable clue."

Overall, understanding open source appears to have required an aggregation of community gossip with available technical information. Since open source involved many players and an array of alternative strategies, putting the pieces together could prove to be an intricate task, as the following exchange about reports on the introduction of a paying service in one of the major component manager indicates. Paul Bristow expressed his concerns about the direction that the Spring component manager was taking:

> “If anything I think I'm more concerned, partly from following this http://www.theserverside.com/news/thread.tss?thread_id=50727 [a report on buying support on Spring to get ongoing bug fixes] before spring source [SpringSource, the developer company behind Spring] had responded adequately (imho) to the concerns expressed.
>
> It still looks to me like they dragged out reassurances about access to the source for branches (I'm still not 100% on whether we have access to fixes that don't apply to trunk).
>
> I think someone in the spring community is going to have to bite the bullet and set up a community distro to maintain synched patched community minor versions the way it's going and am not sure what else this may trigger
>
> Spring seems to be treated as if it's like open source - it's looking to me like some assumptions may need to be reassessed”[168]

In this email, Paul Bristow pointed out a report, which explained the implications of a recently published maintenance policy at SpringSource requiring a subscription for official patches of the code. Bristow went further in teasing out the consequences, and he implied that the decision could result in a vigorous open-source community taking care of their own free updates in a community distribution. He added that it was not clear whether the company would allow that to happen, and if it was not possible, the open source status of the Spring component had to be reassessed. Another Sakai developer responded with news he had gotten of a recently created community distribution:

> “John Lewis pointed this site out to me.
>
> http://www.freespring.org/

> So it looks like someone already has "bitten the bullet and set up a community distro". Not sure what this means for maven dependencies, but it was pretty clear this was going to happen."[169]

The mentioning of the source suggested that it was important for assessing the significance of the new community. In other words, the news was from an authoritative source, which indicated that the new community had future promise. The existence of an open source branch was however not enough reassurance, as it could be partial with respect to dependencies to other components that were important for the Sakai community.

As these contributions indicated, the open-source scene was a landscape in constant movement, with a high level of interdependence across players. Code was implicated in these social processes, which had significance for the direction development was taking. As the previous quote from Clay Fenlason suggests, the choices of others, especially respectable projects, were valuable clues. This attitude was widely shared within Sakai, as Mark Norton's argument in favor of adopting JSR also illustrates:

> "Opinions vary considerably on the importance of support the emerging JSR-170 Repository specification. Likely JSR-170 will prove to be of importance largely in the Java community, and since Sakai is strongly influenced by Java, support for this specification is desired."[170]

This paragraph served as an introduction to a longer document from March 2005, arguing with technical detail on the possibility of the adoption of the JSR standard for a content management framework *before* the standard had been finalized. (The JSR-170 standard was released in June 2005.)

The dynamics of the evolution of software in open source and open standardization communities have been extensively studied, as I have described in Chapter 4. It is not my goal here to contribute to this line of research; the development process of the new Sakai would not provide sufficient basis for this. The point I want to make is that engaging with open source was the epistemic strategy of choice for Sakai back-end developers when it came to mastering the complexity of a computing landscape in evolution. Most importantly, the changing context was taken for granted by participants, in the sense that they were conceiving of their actions against the background of that context. The necessity of being part of change was not questioned or debated, discussions addressed the details of how Sakai should partake of the change. Second, participants looked to the open source scene for making sense of a broader technical ecosystem in evolution, and trusted it for offering back-end solutions in a complex and evolving technological landscape. In this manner, the broader evolution of technology was mediated to the new Sakai in terms of the adoption of open source components, and a related awareness of the open source scene.

The extent and nature of the trust in open source should not be underrated. The frameworks and other components were not simply perceived as technical solutions to technical problems. They were understood to be outlining a solution for what was possible in terms of functionality given the current state of technology. This is clearly what Ian Boston described in his defense of the adoption of back-end components like JCR after the second failure of the platform:

> "The conceptual architecture of the server that sat behind Sakai OAE was solid but it was crippled by two factors. […]

> The evidence for these assertions can be found in the old google groups mailing lists as the server team wrestled to implement unimplementable UX/UI requirements on the chosen architecture. Features that forced the server team to attempt to implement a Social Content Management system using an Enterprise Content Management System (the key word to search for will be BigStore). Features that made the publishing model, successful in so many global social networks untenable. Features that forced the abandonment of hierarchical content structures in direct conflict with the 'teachings' of Roy Fielding, 'God Father of REST'."[171]

Ian Boston was practically saying that JCR was a component that provided a solid implementation of the publishing model on the basis of "the 'teachings' of Roy Fielding, 'God Father of REST'". From his perspective, this was a solid software solution based on authoritative sources of knowledge. This solution made possible a model of interaction: the publishing model. What it was doing was "successful in so many global social networks". In other words, the valuable knowledge of the makers was needed to strike the balance between technical feasibility and functional desirability. The publishing model should have been respected as that which represents a socially meaningful application of what technology can do. Instead of this, designers were disrespectful of the functional approach that JCR made available, and they came up with features that "forced the server team to attempt to implement a Social Content Management system". This line of argument indicates that components *could* be perceived as pre-established packages translating technical opportunity to a functional solution, and the knowledge required for creating them commanded respect.

A similar line of argument was outlined in connection with the adoption of open-source components by Michael Feldstein, a technology consultant and member of the Board of Sakai:

> "One of the more radical departures that Sakai 3 makes from traditional LMS design is that everything in the system is treated as content. A traditional LMS is an aggregation of tools—discussion boards, grade books, test engines, wikis, assignment drop boxes, etc.—each of which has its own data model in a relational database. It's really a hodgepodge of separately designed tools that are knitted together through a user interface layer and a few common services. In contrast, Sakai 3 is being built on top of the Apache Jackrabbit reference implementation of the Java Content Repository (JCR) standard. Everything is treated as content, including grades, test questions and answers, discussion threads, syllabi, personal profiles, chat messages, and so on.
>
> This approach has some benefits in terms of shoring up common weaknesses in LMS designs. But, more profoundly, it also leads to some pretty fundamental changes in the way that learning environments can work, thanks in large part to the strong and growing adoption and maturation of useful content integration standards.
>
> Let's start with the more mundane improvements. There are several areas where, because of their design heritage, LMSs tend to be weak and Sakai 3 can be better:
>
> Search: Developers for traditional LMSs have a very hard job when it comes to designing effective search. […] As a result, most LMSs don't do search particularly well. In contrast, Sakai 3 is being built from the ground up on top of a

> content repository, which was designed precisely to handle search use cases. Theoretically, search in Sakai 3 should be best-in-class.
>
> Tagging: In some ways, this is just an extension of the search problem. Developers in a traditional LMS architecture have a real challenge wiring tagging uniformly into a very non-uniform system. As a consequence, there are lots of places in a traditional LMS where tagging could be useful but doesn't exist. Once again, with everything stored in a content repository, adding universal (and useful and consistent) tagging should be much easier in Sakai 3.
>
> Archiving: A third area where LMSs typically struggle is in mass course archiving, and once again for the same reason. […] On the other hand, if everything in your LMS is stored in a content repository, then you can treat everything like…well…content. Archiving should be much easier, much more performant, and much more uniform.
>
> Those are some of the useful but boring benefits that come from the 'everything is content' architectural approach."[172]

Feldstein suggested that the adoption in Sakai 3 of the Java Content Repository (JCR), and its underlying architectural approach of "everything is content" "led to fundamental changes in the way that learning environments [could] work" in terms of three much needed areas of functionality: search, tagging and archiving. These new functionalities were "the useful but boring benefits that [came] from the 'everything is content' architectural approach." Here again, we see the notion that open-source components provide packaged solutions of functionality. Like Ian Boston, Feldstein was also linking the availability of these solutions to the "maturation of useful content integration

standards", which made available the high-level architectural design work required for addressing content-related functionality. He also emphasized that this high-level design perspective was missing in earlier LMSs, which had emerged as a hodgepodge collection of tools with a data model of their own.

While the Sakai community was inserted into the broad context of evolving computing, it was looking to the open source landscape for making sense of change for its own purposes. Overall, the landscape of open source was perceived as an environment in motion whose contribution was to engage with technological change, both by providing solutions for mastering change and by turning change into opportunity. Open-source components could be viewed as carefully designed solutions whose architectural design made available new domains of functionality. Once the desirability of functionality was assumed, component adoption came to be translated to the technical problem of integration with a software ecosystem in motion. In sum, the broader change of computing became mediated to the Sakai community in terms of the adoption of back-end technologies, and the epistemic processes that this adoption involved. As I have shown above, engaging with open source involved the ongoing awareness of a changing environment on the one hand, and the practical engagement of getting software to interoperate on the other.

Meanwhile, the story of open-source components was not finished with their adoption. Making components available for Sakai contributed to the formation of the open-ended design space I have described. I will now continue with the story of open-source components within Sakai.

9.2.3. Contribution to the design space through promise and validation

Around the time of the formation of the Sakai 3 vision, the JSR-170 specification and its subsequent implementation in the Apache Jackrabbit content repository for storing, searching and accessing content became important drivers behind the design space, creating optimism about a content-based Sakai, and driving much of the thinking about the "Everything is content" paradigm. JSR-170 targeted the separation of the structures of file-storage from the functional structures of content management, making way for the reuse of digital content. Sakai participants started to monitor the JSR specification from the early days of the Foundation, before the specification was even released by the Apache Foundation.[173]:

> "Opinions [within the Java community] vary considerably on the importance of support[ing] the emerging JSR-170 Repository specification. Likely JSR-170 will prove to be of importance largely in the Java community, and since Sakai is strongly influenced by Java, support for this specification is desired."[174]

JSR-170 was considered to be an artifact with potential within the Java community, and as such as a community solution to wide-spread content-management challenges within the Java community. Because Sakai was heavily relying on open-source Java components, it understood Sakai 2 as partaking of the problems tackled by the wider Java community, as well as benefiting from the solutions it was offering. Thus, JSR-170 came to be imported within Sakai as a bundle containing the solution to a problem; while there appeared to be wide-spread familiarity with the related issues within Sakai, and a general consensus about their acuteness for users, I encountered no problem in search of a solution. The formulations of the problem always appeared in connection with the

solution: linked to “JSR”, as the specification came to be referred to within and beyond Sakai, or its implementation in “JCR” (Apache Jackrabbit).

The Content Hosting project, the server-side satellite of Resources, came to describe the need behind JCR in the following manner:

> “As the community starts to explore development of new "content" tools in the wake of the move from CHS to JCR, it's worth bearing in mind some previously formulated general requirements in this domain. […]
>
> Sakai's current "Resources" application attempts to cover two areas of functionality:
>
> 1. Context-specific delivery: Straightforwardly assemble various pieces of (mostly uploaded) digital media of various types on a page. (For example, "Week 1 Readings" and "Reference Images" might be provided to students inside folders in the site's Resources tool.)
>
> 2. Administration and central delivery: Provide "file management" in a central location. Give a central place to find all available downloads regardless of media type or context.
>
> Over time it's become clear that these two goals conflict if forced into a single UI. Digital artifacts need to belong to more than one application context: they need to be attached to emails; they need to be embedded in announcements, discussions, and wikis; they need to be arranged in slideshows and displayed in quizzes. But the current Resources tool UI assumes that Resources "owns" the displayed media. To avoid the issues that come with shared ownership, Resources typically doesn't display "attached" files [i.e. those attached to emails] – but then that

> eliminates any chance of centralized administration and forces users to make extra copies of what may be very large files."[175]

Sakai 2's problem was that file storage mirrored the structure of the user interface: whatever files belonged together from the perspective of the user, like a lesson plan with materials, came to be stored together, as files within a folder. This meant that every time a file was inserted into a new context, represented by the user interface, it had to be copied into the new context. JSR created an abstract layer of file storage, and an API with universal identifiers to stored pieces of content, which allowed for creating references to the same digital resource from various locations and contexts in the user interface without the need of duplication.

JSR may be understood as specifically addressing the principle of non-duplication, which came to be referred to by Sakai participants as Pooled Content. While non-duplication was an important abstraction to support various forms of content sharing, it could bring its own challenges. One such challenge was making sure that distributed users had access to the same form of the evolving content, while changes did not endanger the shared resource beyond what was acceptable. As the community was preparing for the first official release of Sakai OAE, this challenge was addressed as the "one big thing" to work on after the release. The related request described "Four Needs for Expanded Permissions", with four different scenarios where users endangered shared content. I will discuss two in more detail:

> "If we understand it correctly, permission options on pages created in OAE and on files uploaded to it are exactly the same, because both files and pages (or areas) are treated as content items (show up in content searches, etc.). It follows,

then, that if I upload, say, a Word doc that I want other users to be able to edit, I must give members "can edit" permission. However, this also gives them permission to delete the page. We need the ability for someone to share a file with other users in a way that allows them to alter its content, but not to delete it.

Use Case: I am president of a student club. I upload a Word file to the club library that I want to give all club members the ability to edit before I post a final version. Since the club has 300 members, not all of whom I know personally, I would be very sad if one of them deleted the document and all the edits that had accumulated in it prior to that point."

"Currently, an area's permissions can be toggled so that members can edit, which does not allow them to delete, rename or change permissions within the area, but does allow them to add, delete, or edit pages within the area. We would like to see a role that would allow someone to add and edit pages in an area, but not to delete them.

Use Case: I teach an introductory Writing course. I want to set up peer editing groups - areas in which a subset of students in the class are able to post and edit each other's pages. However, I do not want them to be able to delete any pages, because I think it will be valuable for me to see even the false starts they may have made along the way."[176]

The scenarios suggest that users' ability to delete parts or whole of valuable content in its entirety, on purpose or simply by accident, represents a significant danger for the shared resource. This problem can be mitigated by nuanced access rights (no delete privileges), and some form of duplication of the resource (as in versioning). The latter approach

contradicts the principle of non-duplication, and suggests that storage abstractions relying on models that do not emphasize non-duplication can be equally valuable for tackling the distributed collaboration with content.

The problem-solution bundle around JSR, as I have said before, only talked to non-duplication, and the scenarios that were illustrative of related problems; the relevance of other content-sharing scenarios for the community suggested that in JSR problem and solution were indeed bundled, and JSR was a solution which picked up its problems upon entering the Sakai community.

While JSR was clearly preceded by its fame, and it was discussed within Sakai before it was available, the import of the actual artifact did not start from discussions, but a prototype implementation by Ian Boston in the summer of 2006. The episode is described in his blog as follows:

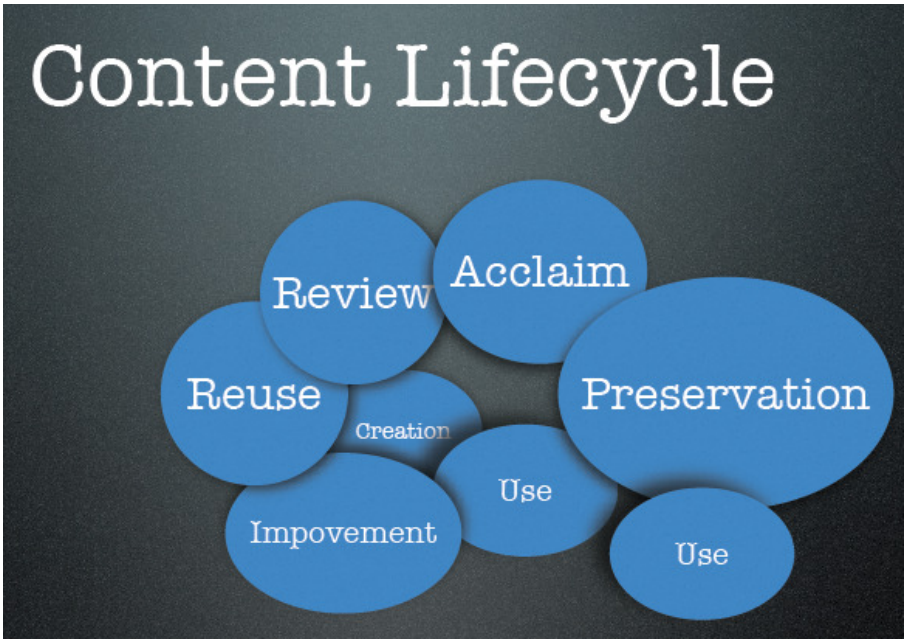
> "Having done a bit of work in Content Hosting already, producing a plugin mechanism as a patch in Sakai 2.1.2 and 2.2, I feel that it has some shortcomings. These observations are not a criticism as it does what is does very well. But I feel that there are some aspects of the implementation that get in the way. For instance, Collections and Resources are separated objects. The storage of objects or nodes within the content hierarchy is done in such a way as to make extension difficult. I have a feeling that the data access patterns are causing performance problems with WebDAV.
>
> Rather than whine about it to the community, I've decided to have a go at using a JCR under the ContentHostingService to re-implement this service. If it works it might m[a]ke life easier.

> Jackrabbit performance looks good and the storage architecture matches what is implemented in Sakai so should work in a cluster. […] The JCR will become akin to an RDBMS [relational database management system], where a back end user acts on behalf of a role in the front end."[177]

Prototyping was presented here as superior to the discussion of problems: Ian Boston stated that he did not "whine" about the problems, but directly resorted to action, to see if the thing would work. From his perspective as a server-side developer, the problem was whether a component technology could be inserted into the existing array of software components in Sakai, specifically to take the place of one particular component, the ContentHostingService. If JCR worked within Sakai, then Sakai could do what JCR was doing. The result of the experiment was validation and promise: JCR could be wired into Sakai, which meant technical validation. Its good performance further validated the effort. This was however not a full implementation, so validation was to be understood as promise of future technical success: JCR "should work in a cluster", it "will become akin to an RDBMS", "where a back end user acts on behalf of a role in the front end". These promises were formulated here in a technical language, but the early JCR validation also gave fodder to non-technical promises. Thus, in a Sakai conference presentation in 2007, Ian Boston suggested a connection between JCR and the following functional aspects of content:

- Support for the "content lifecycle" (see Figure IX.1).
- Variable content creation environments, such as the structured model in publications, the chaotic model in social media, and the collaborative model in education. [178]

Figure 9.1: Representation of the content lifecycle in Ian Boston's presentation about JCR[179]



JCR further came to be introduced to the broader developer community with a hands-on coding activity at a several day workshop in 2008, which engaged participants from four universities in the rapid prototyping of a file-manager application over a JCR-based back-end. The goals and the results of the workshop were described by Clay Fenlason, who was one of the participants:

> "So, one aim of the Cambridge Get-Together was to see whether it would be practical in future to develop some Sakai tools using primarily JSON, Javascript and HTML. Another aim was to tie this in to the work that Ian Boston has been doing on implementing JCR-170 for Sakai.
>
> […]

By the end of 3 days of development, we had a proof of concept "file manager", with the following functionality

- upload multiple files simultaneously into a JCR-170 content repository
- display two different views of the contents of a Sakai site
- create 'Collections' (or categories for grouping content)
- assign files to collections"[180]

In a blog report of the workshop, Michael Feldstein praised the potential of the capabilities that the workshop was illustrating:

> "There's also potential here for much more rapid improvement in Sakai's usability. The Cambridge Get-Together basically took existing content management services and, in four days, built a rich new interface for it that handles fairly complex user interactions. All of the various accounts from various participants mention that the approach really let them focus a lot more on user experience."
>
> The overall assessment was: "All in all, this is very promising work."[181]

In his argument for adopting JCR, John Norman similarly supplemented the technical promises with a functional counterpart:

> "Whole new blocks of functionality may become easily available to Sakai if they are built on top of JCR (e.g. Alfresco Document Management)."[182]

In this manner, JCR outlined a space of promises and opportunities in the area of content. The presentations suggested that it would not only solve existing problems within Sakai, but its solution would open up opportunities in social media and content-driven collaboration, two areas which were of interest for the Sakai community. By the

time these arguments were advanced at the turn of 2007 and 2008, the original experimentation with JCR had grown into a stable component, which had been adopted into versions of Sakai 2 as a contrib tool. The working implementation further validated the promise.

JCR's promise contributed to the emerging Sakai 3 vision, most importantly with respect to the focus on content, and the motto of "Everything is content." This was a phrase borrowed from David Nuescheler, who initiated the Java community process for the JSR-170 specification, and whose company later created the Jackrabbit implementation of the specification. Clay Fenlason described the JCR-based new kernel of Sakai 3 along these lines:

> "A JCR repo is used as the primary data source, in the "everything is content" spirit of the JCR world. The relational DB becomes not so much the data source as the index for JCR content." [183]

In sum, JCR can be pinpointed at the origin of the content-perspective of the emerging design space. At the same time, it has to be added that local domain-driven explorations of content also played into Sakai 3's "Everything is content" approach, which became further colored with the overtones of the web 2.0 movement. JCR may have been the component that got the promise rolling, but the promise itself grew beyond it by the time it became the Sakai 3 vision.

The contribution of JCR to the creation of the design space around the future Sakai is all the more interesting in light of its subsequent demise after the first release of OAE in 2011.

9.2.4. Infrastructural implosion

Server-side development leading to the new Sakai was greatly relying on the adoption of open-source components. Explorations of a grounds-up redesign of the back-end were rooted in a deep-seated dissatisfaction with the limitations of the existing architecture for supporting a more flexible user experience around content. At the same time, open-source components were not assessed in terms of the details of the problem. They were instead adopted in a wholesale approach, where the problems had already been redefined in terms of a creative solution. An important reason for this appears to be the existence of a general epistemic stance among back-end developers, which was looking to the open-source scene for solutions to the broader challenges of an evolving complex landscape of software, and trusted the experience of the community for their assessment and creative solutions. As components came to be pulled into the local settings of Sakai, local problems came to be formulated or reformulated in ways which emphasized the strengths of the solution.

It may be suggested that the local process of decision-making in Sakai with respect to open-source component was aligned with the garbage-can model of organizational choice (Cohen, March, Olsen, 1972). This model was formulated as an alternative to the linear account of the problem-solving process, where first a problem is clearly understood, and then a solution is elaborated in light of the details of the problem. The garbage can model instead posits a messy space, where solution choices and problems exist alongside each other, together with decision-makers and decision situations. Metaphorically speaking, these elements all float within the space, looking to find each other:

> "an organization is a collection of choices looking for problems, issues and feelings looking for decision situations in which they might be aired, solutions looking for issues to which they might be an answer, and decision makers looking for work" (Cohen et al., 1972: 2)

Sakai 2 had issues. The open source community had solutions. Ian Boston was looking for work, like many other back-end developers. The separation of the kernel, alongside the vague talk about a new version of the Sakai platform, created a decision situation related to back-end technologies. In the sakai-kernel list, these elements all found each other, and a decision was made about a novel backend.

The garbage can model could work as a precise account of how the decision on a new kernel emerged out of the conglomeration of the various elements, rather than as the elaboration of a solution to a well-established problem. At the same time, the model would miss the contribution of the adoption process to the vision of Sakai 3, and the emergence of an open-ended design space, which in turn fueled a local process of expansive innovation. This is the process I suggest to call infrastructural implosion, on account of the lines of forces that drive the process of local expansion from the outside in.

There are two significant aspects of the process of component adoption which are relevant for an account of infrastructural implosion: the epistemic stance toward open source, and the tinkering implementation process. The epistemic stance toward open source was based on the notion that open source was a superior distributed approach to mastering the evolution and complexity of software by means of the creation of modular, component architectures. This resulted in the understanding that the generic capabilities

of the components emerged from an architectural model which translated what was possible in technical terms. The adoption of components took the form of a practical process of implementation, which was necessary for validating the possibility of their integration in an existing or desirable ecosystem of software. This process resulted in proof of concept implementations, which were indicative of future technical success. Meanwhile, proof of concept implementations were also seen as indicative of future success in terms of functionality. Proof of concept implementations served as tangible validations of their promise, which worked on two levels: on the technical level, as the promise of a working system, and on the functional level, as the promise of new areas of functionality. In this manner, new components contributed to the creation of an open-ended design space, which prompted participants to make sense of the promise that software components helped to spark.

Based on ethnographic research with web developers, Bucher (2013) described a similar process in connection with the web-based infrastructure offered by the Twitter API:

> "We could say that APIs have opened up the playing field of software development, in the sense that their presence has allowed for a much broader range of actors to use and repurpose the data and functionality of an existing service. Jack, the CEO of a major social media data reselling company, described how APIs have 'yielded the next wave of Internet related innovation':
> *There's been a huge wave of "just open it up and see what happens" that we're just at the beginning of understanding. The implications of which will shudder some businesses, while allowing some to flourish. What the underlying API*

> *supports, or doesn't, indeed is defining social media as we know it. APIs define what we can build, policy-wise, as well as technically, and subsequently the products we build/use/consume, which in turn obviously affect culture and socialization in general.*
>
> Perhaps more than anything else, Jack's comment illustrates how APIs not merely have the power to regulate access to the content of databases and software functionalities, as they currently exist. Rather, APIs are also future-oriented. The kind of 'openness' invoked here, is not one of access or 'free', but of anticipation. 'Just open it up and see what happens' I would argue, signifies a certain kind of openness towards the future, where APIs are essentially deployed to ask developers to reimagine existing services and to transform them into new realities."

In her analysis, Bucher suggests that the Twitter API is an architectural solution of 'programmed sociality' (2012), which has the power to regulate access to data created within the social platform. At the same time, APIs urge developers to reimagine the limited range of opportunity that they make available, and to transform them into "new realities." The context of reuse in the case of the Twitter API is different from the Sakai case of open source components in that technical patterning operates at the level of data exchange, while in the case of open-source components, architecture is replicated by the replication of software. At the same time, both cases indicate a process where software architecture calls forth an open-ended design process which seeks specification in terms of the formulation of meaningful human activity supported by software.

9.3. Discussion

9.3.1. What is Sakai 3?

In the above, I have described a process of development where the import of artifacts with generic capabilities called forth the innovative specification of these broad capabilities. This leads to the question of how we can make sense of the resulting artifact.

Sakai participants talked about the Sakai 2 Content and Learning Management System as a collection of tools around a course site. The actual sites would be created as variations over the template of the tool-based site. Sakai participants were discontent with Sakai 2 because the boundaries of the individual tools had been drawn along the lines of the underlying software architecture. The only reason for the separation of assignments from grading was that the two had been created at different times by different teams. Also, blog, forum or wiki could not be assignments because all of these were separate tools. This led to a questioning of the very notion of tools, and the particular way of bundling the interactive features they represented. The "Everything is content" approach of Sakai 3 replaced tools with widgets, which provided a new model of bundling basic features of interaction. As one participant described the future platform:

> "Sakai3.0 = K2 + social network support from opensocial + broken-down Site silo system permitting more content sharing and open content + flexible way of constructing pages/spaces for a site with widgets and edit-in-place"[184]

A similar description of Sakai 3 was given in a retrospective critique of its development approach:

> "That it's attempting to build a generic platform on top of other generic platforms (Sling, Jackrabbit, etc) may be part of the problem."[185]

These snapshots suggest that web-based software systems like Sakai 2 and Sakai 3 are based on a systemic arrangement of 'patternings' with repetitions and variations, which are aligned with the architectural makeup of the platforms. The generic patternings will be shared across a broad range of sites, where the platform is used, producing an effect of seriality. In the following, I will take a look at the broader sources of how seriality is created in networked software systems from repetition and variation.

On account of the digital character of software, identical copies are made easily. In the open source logic of distribution that the Sakai systems follow, any person or educational institution can download the software code, and thus make a copy to be run on a local server. According to the Sakai Foundation website[186], more than 350 organizations run a copy of a version of the Sakai 2 system.

At the same time, local instances are not identical. Fleck (1993; 1995) coined the term configurational software to describe the approach software development which builds the possibility of local adaptation into standardized software products, and which has become prevalent by the last decade of the 20th century. Relying on an analogy borrowed from engineering, Szyperski, Gruntz and Murer (2002) describe the two faces of software as plan and instance to make the point that the various instances of actual runs of the software can be of different shapes because of local parametrizations and local data. Overall, variation in interactive software comes from two sources: configuration and data. Every instance of a platform like Sakai can be adjusted for a local context through configuration and coding, and they rely on their local ecosystem of data.

The Sakai platforms are also web-based systems, which run in a server-client architecture. This means that a user can interact with the system without having copied

and installed it themselves. Indeed, the actual use of the artifact happens at the clients' end. Most typically, a browser will be needed for using the system, and serving up web pages and other content to users through the browser can be stated as the primary purpose of the Sakai systems. The client has access to these systems by proxy of the Internet and various related software artifacts. The software interface that an end user interacts with on the client side represents a new level of variation over the system installed on the server. End user personalization is again based on configurations and an ecosystem of data. By the time the system is instantiated in use, it has been turned into a very individualized artifact. What remains the same across various end users of an installation is a shared template of presentation and interaction.

In light of the above, tools and widgets, course sites and group dashboards emerge as pre-established templates, which are supported by the underlying architectural models. The widget-based infrastructure of Sakai 3 was made possible by the combination of various standards: the JSR-170 content management model and the RESTful addressing of resources on the server-side, and the JSON, HTML and Javascript specifications on the client side. Taken together, these standards made available the generic template of a widget-based approach, which came to be specified in the final design of the software. While the group dashboard was a novel conceptual construction, other outcomes of the widget-based approach, like pages of mashed up content or the site-wide availability of a chat-widget may be seen as following established user interface models based on these standards. Meanwhile, the overall combination of the various functionalities within Sakai 3 resulted in a very peculiar systemic artifact, which emerged as a collection of variations over the generic capabilities provided by its architectural foundations.

In this sense, I suggest that S/OAE as a complex systemic artifact may be understood as the result of a specific design process over the range of adopted software technologies. Much of this design process produced conventional, standard forms that had been used in web-based software in many instances before S/OAE, while a small section of it resulted in conceptually novel constructions. The latter version of the design process is what I have described as infrastructural implosion. I will now expand this definition, and suggest that we may understand infrastructural implosion as the process which is called forth by the adoption of infrastructural technologies, and which results in a mix of routine and innovative solutions that are specified on top of the generic capabilities of the adopted foundations. Sakai 3 as an artifact may in turn be understood in terms of the concrete process of infrastructural implosion which gave rise to it. I have shown that the group dashboard incorporated conceptual connections to the social world of higher education, but it had no direct counterpart in terms of established activities or social situations.

9.3.2. Infrastructural implosion as an alternative evolutionary account of technology

I have started this chapter by suggesting that the evolving landscape of networked software has received scarce attention from human-centered approaches to software. I have outlined infrastructural implosion as an account which frames the expansive design process of Sakai 3 within the broader context of this evolving landscape. I will now argue that this process stands apart from social models of technological evolution.

SCOT has provided two major models to account for the social construction of technological change: Bijker and Pinch's model for the evolutionary closure of technical artifacts, and Hughes's account of large technological systems. The latter model has been

recently extended by Egyedi to make sense of the bottom-up systemic growth of inverse infrastructures.

Pinch and Bijker (1987) have suggested that the task of social constructivist studies of technology is to explicate the evolutionary closure of technical artifacts (see also Misa, 1992). Evolutionary closure implies a process of change to completion: the artifact's growing into a consolidated state of recognition, where it is both acknowledged on its own right and distinct from other artifacts. Bijker and Pinch's concept of closure is inherently linked to a technical ecology where artifacts are manufactured as batches of roughly identical series. Closure is the point when production keeps repeating a similar design over time, and the design may spread to various hubs of manufacturing. This ecology (and economy, or organization of labor) produces a repetition of sameness. Bijker and Pinch argue that closure is the outcome of a series of episodes in the interpretative evolution of an artifact, during which the current design of the artifact becomes reinterpreted in use by relevant social groups, and the new interpretations are fed back to further designs, until the form of the artifact reaches closure.

Thomas Hughes has used an analogy from mechanics to describe the evolution of the network of electricity as having mass, velocity, and direction, which contribute to its momentum (1993). Momentum implies that technological systems not only set limits to interpretive flexibility, confining how they can be developed, but they also shape society by exhibiting a presence that demands to be acted upon (Hughes, 1994). Momentum results from the financial investment in the installed base, the vested interests of professional groups familiar with the technology, and the large administrative bureaucracies that grow around such systems (Hughes, 1987; 2005). These socio-

technical forces arrange the social world around technical systems according to the systemic logic of the latter, and result in the overall expansion of the technical system itself. The 20th century in particular has been characterized by the emergence of complex and large technical systems, described as "spatially extended and functionally integrated socio-technical networks" (Mayntz & Hughes, 1988), such as electricity, railroad and telecommunications. Studies of information systems, such as Bowker and Star's (2000) analysis of the growth of the International Classification of Diseases, have documented a similar process of expansion, which parallels Hughes' account of the evolution of large-scale technical systems in that it has involved the projection of the classificatory logic to new areas, and the incorporation of existing local systems of classifications.

This particular pattern of systemic growth in large systems has been characterized as top-down, with reference to the importance of some sort of central coordination and ownership. Recent studies of infrastructure have turned toward the growth of decentralized, self-organizing technological systems, dubbed inverse infrastructures, of which the Internet appears to be a prime example (Egyedi, Vrancken, Ubacht, 2007). Further examples include Wikipedia, P2P networks, privately owned solar energy networks and urban WiFi (Egyedi & Mehos, 2010). A central feature of inverse infrastructures is horizontal coordination between actors for the creation of the system, which is in contrast with the vertical, top-down coordination characteristic of the types of large technical systems described by Hughes. Large-scale technologies perpetuate their own systemic logic in a top-down manner, by the weight of the installed system. Inverse infrastructures grow bottom-up from cooperation. Meanwhile, the motivation for

cooperation in inverse infrastructures is still the creation of an installed base that has the momentum to thrive.

The process of infrastructural implosion differs from the above models. It combines the pattern of systemic repetition of functionality characteristic of large scale systems with episodes of reinterpretation, which occur when systemic components are taken up in the course of design. In this process, the repetition of large scale systems happens as an infrastructural implosion, which specifies new variations of their generic patterning in a creative manner.

Hanseth and his colleagues have described a different type of infrastructure, which they have called Information Infrastructures (IIs) (Hanseth, Monteiro, Hatling, 1996). IIs have the characteristic of infrastructures previously described by Star and Ruhleder (1996), in that they stretch across space and time: they are used across many different locales and endure over long periods. IIs are further characterized by:

- openness in terms of the type of users they can accommodate;
- the combination of a multiplicity of agendas, purposes or strategies by means of the interconnection of numerous modules or systems;
- dynamically evolving portfolios of an ecosystem of systems;
- the accommodation of an installed base of existing systems and practices.

In a recent article, Monteiro, Pollock, Hanseth and Williams (2013) have further argued that the systemic characteristic of IIs was important for understanding their evolution. Most importantly, they have suggested that standardization and the embedding of one technology with other, apparently unrelated modules act as systemic constraints on the evolution of these infrastructural software systems. In line with this, the design of IIs has

been described as a process of generification, which results in the alignment of specific use cases with systemic constraints.

Sakai 2 and 3 share the characteristics of IIs as described by Hanseth, Monteiro and Hatling (1996): they have been conceived to accommodate a range of users, who bring multiple and varied agendas to the system; they bring forth a collection of functionalities; they are inserted in a local ecosystem of technologies and related practices, and their modular architecture aims at the possibility of dynamically evolving portfolios of an ecosystem of modules. At the same time, I have shown that their design has followed a process opposite to generification: infrastructural implosion may be understood as a process of specification over a modular back-end supporting generic capabilities. I will further add that the evolutionary model of information infrastructures do not reflect on broader context of the evolving computing landscape, and its role in the evolution of IIs.

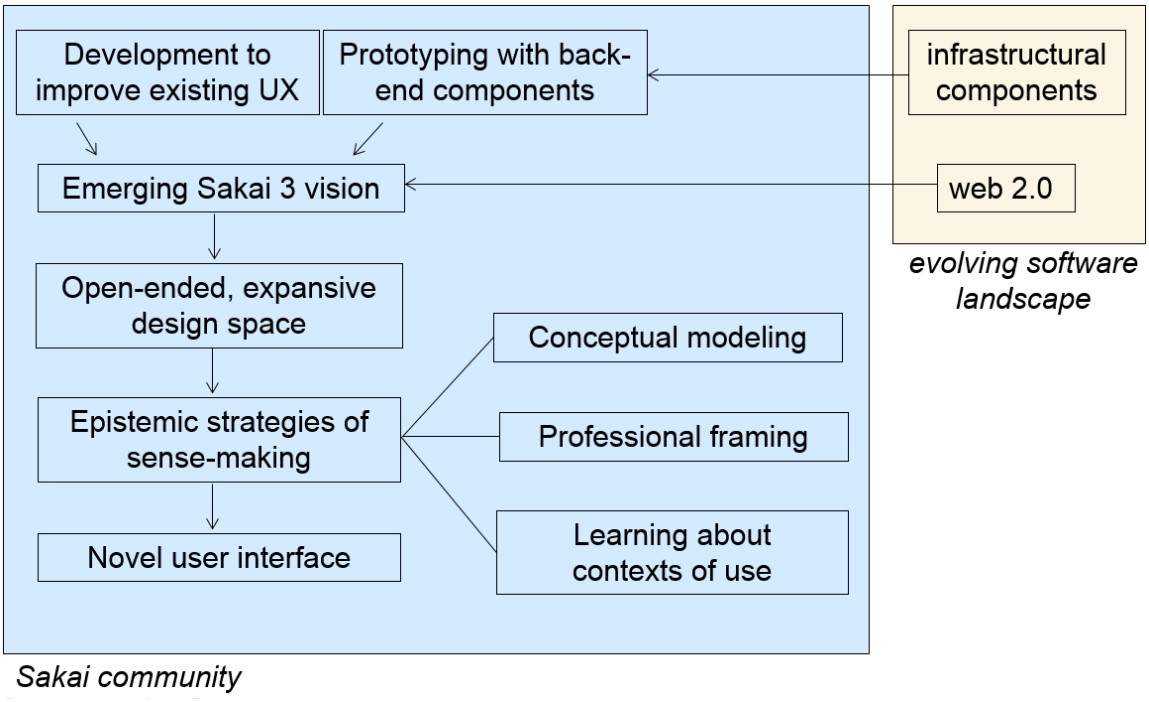
In light of the above, the process of infrastructural implosion may be seen as a distinct model of technical evolution characteristic of networked software. I have characterized infrastructural implosion as a design process emerging in a context of technological change, and I have also suggested that it brings about a combination of systemic repetition and interpretative specification, which results in bubbles of innovation in a process of systemic functional patterning.

CHAPTER 10. DISCUSSION AND CONCLUSION

In this chapter, I will discuss the findings of the analysis from a comprehensive perspective. I will revisit the case of Sakai 3 to present the analysis as a coherent narrative, and I will discuss the theoretical implications with regards to social historical studies of technological innovation and human-centered computing in connection with this overarching narrative. This discussion will be followed by an account of the limitations of the research, and the main contributions. I will close the chapter with implications for research and practice.

10.1. Discussion of the analysis of the case studies

Figure 10.1: An outline of the making of Sakai 3



I will start my discussion with a broad outline of the making of the new Sakai platform. Figure 10.1 provides an outline of my account of the process. Sakai 3 story began with Sakai 2, in a project for improving user experience around content in Sakai 2 (Development to improve existing UI in Figure 10.1). That project ran into limitations in the back-end. Around this time, new architectural and coding specifications were becoming available in the open source community for a more flexible approach to content management. Open-source components were taken up in prototype implementations within Sakai. In the MySakai project, Sakai developers experimented with a separation of back-end from front-end, relying on Ajax, Javascript and JSON in combination with HTML on the front end, and on the Jackrabbit content management platform on the back end. Figure 10.1 shows that 'Prototyping with back-end components' was relying on 'Infrastructural components' becoming available within the broader context of an evolving landscape of software. The success of the early prototype implementations indicated the viability of the approach, and they gave a broad hint of what may be achieved with the new open source software components in terms of functionality. Sakai members started exploring the possibilities indicated by the prototypes, and in the course of 2008 the vision of a new system distinct from Sakai 2 emerged within the web of discourse (see 'Emerging Sakai 3 vision' in Figure 10.1). The executive director gave a coherent formulation to the vision of Sakai 3 at the end of 2008. His Sakai 3 vision pulled the existing threads of work together under a web 2.0 umbrella As Figure 10.1 indicates, web 2.0 was another source of external inspiration originating in the evolving landscape of software. The formulation of the Sakai 3 vision resulted in an open-ended design space, shown again in Figure 10.1. The vision invited participants

to translate the general vision of web 2.0 to the local context of higher education, which could be pursued in the design space by taking further the early promising prototypes. This resulted in efforts of sense-making that I have described in my analysis as conceptual modeling, professional framing and learning about the contexts of use (listed in Figure 10.1). Right after the formulation of the vision, the first true Sakai 3 project was started. This first project was subsequently followed by several others, which included mainstream development efforts focusing on the making of the new platform, as well as side projects, which were seeking indirect and long-term influence. Finally, in the summer of 2010 a so-called managed project was created, which basically ended the actual design of Sakai 3. The pre-release of the new platform came out in October, under the name of Sakai Open Academic Environment, and it was piloted at one of the participating universities. The first official release Sakai OAE came out in the next year, and it was again piloted at the same university. Both of these releases ran into serious performance troubles, which resulted in a general overhaul of the platform's back-end. While the system was struggling with performance challenges, the coalition of participating institutions also came apart, leaving Georgia Tech and Cambridge as sole partners in development. The new Sakai eventually became incorporated into a new foundation, and it has continued to improve under a modified agenda. On account of these more recent events, many Sakai participants look back upon Sakai OAE as a failure.

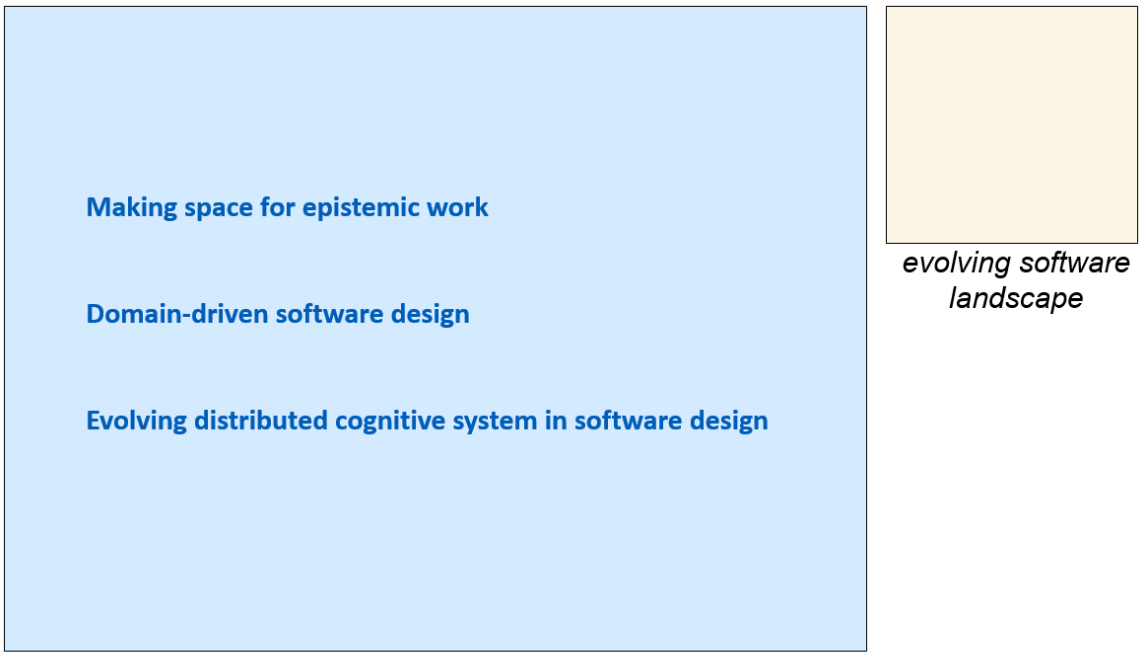
10.1.1. A distributed cognitive account of the domain-driven design of a software system in higher education

I have argued that the Sakai community was created in response to the way change management was handled by large commercial providers of educational software platforms, like Blackboard, which basically imposed a pace of change for the adoption of new releases, while leaving only marginal space for institutions to shape the direction of development. Partner institutions created an open source community for their employees, who were in strategic organizational positions for connecting local educational practices with software technology. My analysis has described the creation of Sakai's open source community as a forward-looking strategic move to *make space for epistemic work*. Central to this was the goal of making space for *domain-driven design in the open source context*: accommodating perspectives rooted within the domain of higher education in the processes of open-source software development. Open-source was known to have little interest in design and requirements, and before Sakai, participants in higher education did not have much say in the design of educational platforms, so a central challenge of the Sakai community was making space for domain-driven design within open source. The Sakai community was put together so as to allow for channeling the knowledge of instructional technologists directly to the design of the software platform. Figure 10.2 indicates the contributions of my analysis to an account of the making of the open source community of Sakai.

I have also argued that the community of Sakai may be understood on various levels as a response to the evolving software context by means of epistemic strategies relying on distributed cognition. The strategies created the space for epistemic

contributions to the design of software by distributing the process temporally, socially, and across artifacts. I have argued that the resulting open source software development community could be understood along the lines of Nersessian and her colleagues' DCog analysis of engineering labs as an *evolving distributed cognitive system* (Nersessian et al, 2003) (see Figure 10.2).

Figure 10.2: Overview of Sakai 3's design process, part 1: making the open-source community of Sakai



The design of work practices within the community was relying to a great extent on open source practices of modularity and web-based communication. Modular software architecture and web-based, distributed communication platforms were used in a strategic manner to make space for software development on epistemic terms. In the case of Sakai

3, modularity was a means of supporting a division of labor that could accommodate back-end and front-end software skills on the one hand, and local development projects defined in terms of local interests on the other. The Sakai community also adopted a range of open source practices to engage with the evolution of software. Open source strategies of release management and prototyping were the most important among these. With these practices Sakai also adopted an outlook on software which was based on the notion of evolution.

Related to the extensive use of web-based communication platforms in open source, Scacchi (2002; 2009) described the emergence of a web of discourse, where features of new software become defined in an unplanned, distributed process. My analysis of the web of discourse in Sakai found support for Scacchi's claim about the significance of online communication for the design of software, but also suggested that the seemingly spontaneous and often chaotic processes of the web of discourse could be understood in terms of local epistemic strategies of a temporal character, which participants adopted to engage with technological change. I have pointed out how framing, anchoring and staking were used in attempts to provide continuity and direction to a shifting collaborative process of development. Collection was commonly initiated by participants as a means of pulling in and managing knowledge about the domain, and it was commonly relying on framing, anchoring in prior development efforts, and staking for future continuity.

My analysis has further suggested that explorations of opportunities within the Sakai community were significantly indexed to the broader evolving software landscape through infrastructural back-end solutions. This points to the emergence of a broad

distributed cognitive environment in open source, where a group of developers with specialized knowledge become entrusted to produce back-end components that work with cutting edge software technology to address pressing problems perceived within the community. The local adoption of novel architectural models was an important local driver for change. I have argued that prototyping activities presented an early validation of the promise of new server-side approaches, and projected the possibility of a novel system.

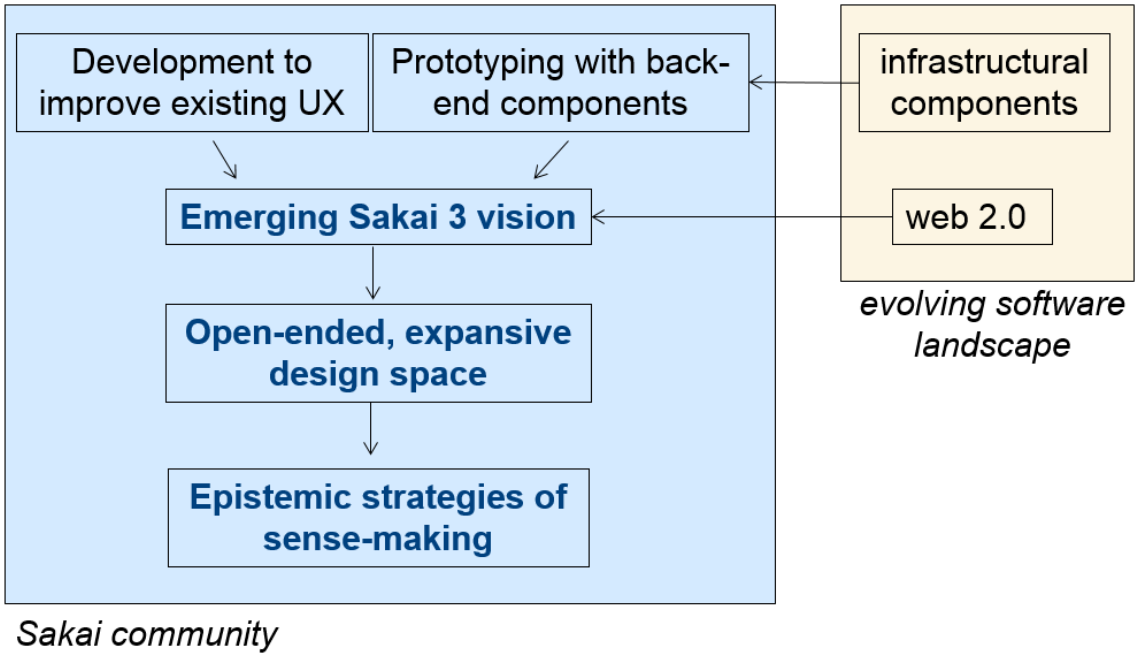
Sakai 3 emerged from two threads of activities that were engaging with the evolution of software. On the one hand, there were a series of projects which were seeking to improve the user experience of the existing Sakai 2 platform, and on the other, there existed a line of prototyping efforts which were implementing novel open-source back-end technologies in Sakai 2's local software ecosystem. The possibility of a new system gradually emerged from these dispersed activities, which provided the foundations for formulating the vision for a future Sakai platform. While efforts at improving the existing system yielded a rich sense of the desirable features and the related challenges, prototypes of the back-end provided tentative manifestations of a future Sakai, which represented a promise for the new system and a validation of that promise at the same time.

I have shown that the *Sakai 3 vision* led to the creation of a *distributed design space* by creating an abstract coherence around existing threads of work in terms of a future artifact based on web 2.0 principles. The creation of the design space for Sakai 3 could be seen as another instance of an epistemic strategy whereby participants collaboratively made space for future epistemic work. The making of the design space

conveys the idea that participants go about designing software by creating the conditions of their own distributed epistemic work of design.

I have argued that the design space unfolding from the Sakai 3 vision prompted a generative process of conceptual construction, which was distributed in time, among participants, and across humans and artifacts. The Sakai 3 vision led participants to make sense of existing general purpose prototypes and generic web 2.0 user experiences for the purpose of the educational domain. Figure 10.3 outlines the influences in the formulation of the Sakai 3 vision and the resulting design space.

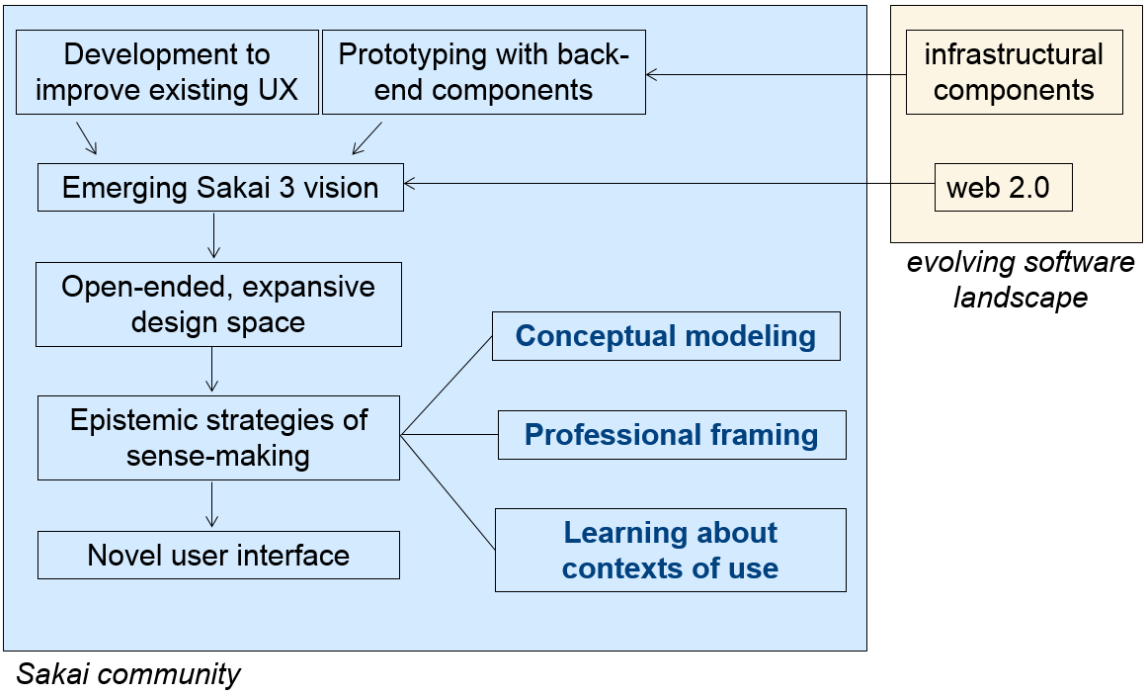
Figure 10.3: Overview of Sakai 3's design process, part 2: the creation of an open-ended design space



The open-ended design space prompted participants to attempt to make sense of web 2.0 for the purpose of higher education. This resulted in efforts of sense-making that

I have described in my analysis as conceptual modeling, professional framing and learning about the contexts of use (see Figure 10.4).

Figure 10.4: Overview of Sakai 3's design process, part 3: epistemic strategies in the open-ended design space



In trying to improve the user experience in Sakai 2, participants came to the conclusion that the main problem was the unnecessary coupling of significant categories structuring the user experience: course sites were coupled with tools, with content, as well as with groups of people and their roles. The prototypes represented a promise of going beyond these couplings. In the first Sakai 3 project, the focus was on just content, and groups were set aside for the sake of simplicity. At the same time, in discussions in the web of discourse, participants were spontaneously coming back to the problem of groups, trying to connect the different conceptual elements in a meaningful way. I

described this as a process of *distributed conceptual modeling*, relying on thought experiments. The model of a new group-based dashboard interface emerged over a period of several months from a number of scattered episodes, as participants attempted to make sense of groups in line with the broad directions outlined in the Sakai 3 vision (see Figure 10.4).

The modeling started with individual models, but it became distributed across participants because models were collaboratively explored in the web of discourse. The various models were described for the purpose of sharing, and they were scrutinized by others participants. The suggested models were examined for their conceptual connections with participants' knowledge about Sakai 3 and higher education, and they were pitted against particulars of education with the help of imaginary scenarios, which were formulated to tease out the implications of each model. The local episodes of conceptual modeling resulted in a series of parallel but contradicting models, which eventually gave rise to a novel group-based model of the user interface. Groups were not to be presented based on content, like a course site, they were to have a profile page containing a history of recent activities associated with the group in the form of a dashboard with activity feeds. This approach allowed the dissociation of content from groups.

The emergence of the novel model of the user interface was a lengthy process distributed over the timespan of several months. Most importantly, the examination of suggested models was not conclusive, local discussions emerged and then they were abandoned. Meanwhile, participation in discussion brought a slow accumulation of ideas, and familiarity with various models. This allowed people to eventually break out of their

initial perspective, and formulate a novel model. The new model was independently formulated on two occasions, and it was subsequently consolidated and accepted by the community in further formulations.

Overall, the front-end development of Sakai 3 was strategically using the professional practices of user-centered design in the same way that back-end development adopted open-source practices of software development. The UX-centered design approach was an epistemic strategy to make space for the domain in the design process. My analysis has shown that it successfully created a *professional framing* around the conceptual processes of design in terms of user experiences. User interface prototypes played a significant role as *framing devices*, projecting a conceptual coherence for the future artifact in terms of the user interface. In this manner, prototypes became implicated in the conceptual processes in an indirect manner, as generic exemplars for what was being modeled.

The analysis of the design process of Sakai 3 also indicated that user activity was only one possible focus in domain-centered approaches. In this respect, it may be understood as a professional epistemic strategy that made selections over the realms of knowledge that could be brought to bear on design. These selections did not simply inform the conceptual process of the design of the software platform, but they were informing its generativity, lending direction to conceptual expansion in an open-ended design space.

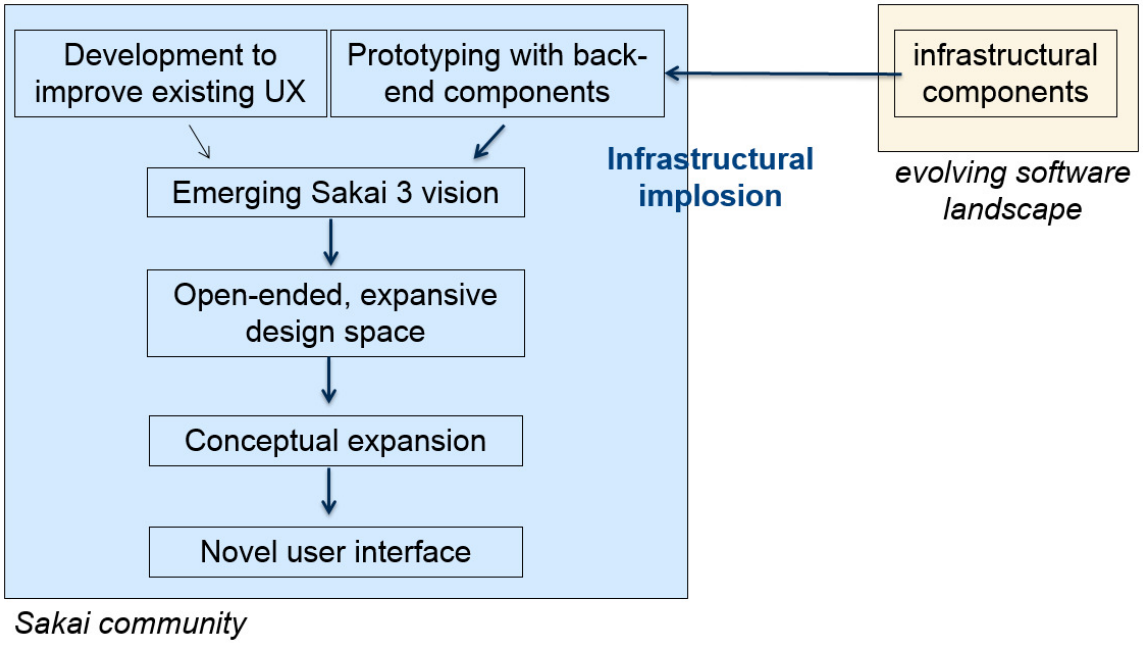
The conceptual struggle to make sense of the new interface propelled participants to bring in new understandings of the contexts. This gave rise to *epistemic strategies of learning*, which partook of Sakai's broader strategies for knowledge management, and

were thus socially and temporally distributed in nature. I have argued in particular that the creation of the Sakai community made way for long-term strategies for *mediating knowledge about the contexts of use* by means of the collaborative production of a knowledge-base which would be available for informing the efforts of design. This was a distributed cognitive strategy, which was relying on a combination of all three facets of distribution: participants were gaining expertise over time; they were sharing their personal expertise in local efforts of collaboration; and finally, the communication platforms acted as external memory for storing the knowledge gained for local purposes, which could be subsequently reused by others. In the overall distributed process, participants were creating a shared outlook on the educational domain through the collaborative enrichment of conceptual models. Following Shore's suggestion to understand a culture as a large collection of heterogeneous but overlapping models (Shore, 1996), the result of this process may be seen akin to the making of a local culture. This culture was partaking of the educational domain, but it also instituted a reflexive edge over the practical world of higher education, which encouraged the collaborative production of conceptual accounts of practical engagement with software as a means of epistemic reflexivity.

To complete the account of Sakai's design process, I will now return to my starting point, the challenge of managing a changing software context, which is often referenced with Moore's law. The process of design that I have outlined could be seen to be embedded within an evolving landscape of software. I have shown how the local adoption of an evolving pool of component technologies with generic functionality resulted in local processes of innovation. I have proposed to call this process

infrastructural implosion. Infrastructural implosion consists of the spread of a generic pattern of functionality to local contexts, followed by local bubbles of innovation on top of this broad functionality. Local processes of design specified the infrastructures in an expansive, innovative manner. In this manner, infrastructural components created a local design space where they exploded inwards (see Figure 10.5).

Figure 10.5: Overview of Sakai 3's design process, part 4: Infrastructural implosion



10.1.2. Replacing social distribution of interpretations with the social distribution of cognitive process

I have argued that social theories of change in technology and software formulated a procedural account in terms of shifting social configurations of interpretations, where dynamic social processes were cast over a set of static semantic constructs. I have described a distributed cognition framework to expand this approach into an account of a

socially and materially distributed cognitive process. I will now summarize what may be gained from this approach.

I have described a process in which conceptual novelty is the result of *cognitive-epistemic construction*. As Nersessian has argued (2005), the cognitive account links the outcomes of an innovative design process to the capabilities inherent in the mind. My analysis has focused on the generative character of conceptual modeling which produces novel conceptual structures from existing ones. This may be described as a process of cognitive bootstrapping, where novelty is projected and systematically explored in imaginative forms of thought. These cognitive processes do not simply reshuffle existing semantic contents, as a social constructionist account would suggest; the human mind is the center of a distributed process which creates genuine novelty. I have proposed the metaphor of *directionality* to make sense of how existing conceptual structures inform conceptual expansion in an open-ended bootstrapping process.

While the mind could be seen entangled in its own meaning-making logic, the distributed cognitive framework further suggests that humans reflexively create the conditions of their own epistemic contribution by creating the environments in which they exercise their cognitive capabilities. I have described a range of *epistemic strategies* that participants devised to further their own capacities for understanding in their local context (see Figure 10.4). I have suggested that epistemic strategies effectuate epistemic selections which lend new directionality to the process, without defining its outcomes. Thus, while social constructionist accounts operate with a static concept of frames, which portrays human groups as passive bearers of interpretative perspectives, the distributed

cognitive approach suggests a dynamic account of framing, in which humans reflexively shape their epistemic contributions to innovation.

Epistemic space is another concept that has been introduced to convey the non-definite directionality of the cognitive-epistemic process of software design. We may think of the making of epistemic spaces, including the design space, as a way of lending directionality to design. Spaces provide directionality, without finality. In this manner, humans are able to shape artifacts by shaping the direction that the design is taking. The concepts of space and directionality together outline an account of *agency*, or how humans can make a difference in technological change.

Taking the distributed cognition account of design as the foundation of our understanding of human agency implies that we adapt our notion of agency to our understanding of the distributed powers of the mind. The notion that the mind is able to devise epistemic strategies that create the conditions of its own creativity are central to this understanding. Besides the creation of a design community and the formulation of the design space, the epistemic practices of the professions may be counted among these strategies. I have suggested that the contribution of these epistemic strategies to design may be understood in terms of directionality. This implies an open horizon of meaning-making, to which epistemic strategies give conceptual direction, without being able to define its exact path. The actual processes of conceptual construction may result in novel conceptual models, which partake of the general direction of thinking, but take it further to innovative forms. What matters for the directionality of innovation is the domain of conceptual understandings that have become productive in the process of design.

Epistemic strategies result in a procedural logic of distributed cognition, which can be understood as broader patterns of socially and materially distributed configurations of cognitive processes unfolding over time. My analysis has highlighted two levels of procedural logics in the design of software, which were layered onto each other:

(1) I have described how Sakai participants engaged with an evolving landscape of software in a prototyping process based on hands-on engagement with software artifacts. I have further described how an *open-ended design space* was constructed from these foundations.

(2) I have described how a combination of professional practices with open-source community building resulted in a *software development community*. Sakai's development community was relying on socially distributed knowledge management practices which were framed in terms of a domain-driven and user-centered approach.

We may speculate that a variation of these procedural logics, and the epistemic strategies behind them, could be identified at other sites of software development, as a common response to the evolutionary landscape in software. I have also suggested that they may be used as the foundation for outlining alternative strategies, such as the social-technical imagination.

10.1.3. Social-technical imagination within a distributed cognitive process of design

Related to the role of professional practices I have argued that the performance failures of Sakai may be interpreted as a lack of social perspectives in the design process. Piloting Sakai with real users in the educational context resulted in performance troubles in two

subsequent releases of the system. It became clear related to these pilots that the back-end was not prepared for handling the patterns of use, which were characteristic of higher education. The assessment of piloting troubles resulted in complex conceptual models of the platform which connected software architecture with patterns of user activity characteristic for higher education. These conceptual models emerged retrospectively after the platform was released, but they were not present during the process of design. Thus, it may be said that the Sakai 3 platform was not prospectively designed from a social perspective because it was not imagined in a tentative manner in terms of patterns of user activities. What's more, there appear to be no professional practices at hand that could be adopted to guide design with respect to the social patterning of use. Related to this, I have suggested that there is an epistemic gap in the social-technical design of software platforms, which points to the need of devising epistemic strategies that can make space for the *social-technical imagination* in design. I have also suggested that these epistemic strategies should be informed by the example of user-centered design, and more generally by the distributed cognitive account of the design of the Sakai 3 platform.

In light of what has been said about the generativity of knowledge, and the expansive nature of the design process, an important goal in *making space for the social-technical imagination in software design* is to make social and sociological understandings of the contexts of use productive in the process, so that they can contribute to the direction of conceptual novelty.

The case of Sakai 3 may lead us to speculate that social-technical design is significantly connected to back-end technologies, and the general architectural

foundations they provide to the software system. These architectural frameworks in their current form may be said to implement abstract principles in complex technical format, the engagement with which requires specialized and advanced knowledge of software technology. Related to this problem, I have also made the point that framing devices could be central for social-technical imagination. User-centered design was shown to rely on prototypes, which are tentative, but fragmentary forms of the artifact. At the same time, getting a sense of the patterns of use from prototypes may still require considerable mental augmentation, and the technical format of server-side technologies promises few hints and cues for that. Thus, I have explored examples from game design to suggest that social-technical design may need special framing devices, which support the conceptualization of patterns of use in terms of significant architectural aspects of the underlying software, and may also allow for dynamic engagement to explore different scenarios of use.

The distributed cognitive analysis of software design implies a reconfiguration of analytic knowledge and knowledge about the contexts of use as it has been generally outlined in theoretical accounts of human-centered approaches to software design. These accounts emphasized empirical contributions, implying an empirical nexus to a social reality, and an understanding of software design as the fitting of software to this (often constraining) reality. My account of the distributed cognitive practices of software design has highlighted a process of conceptual construction which works with conceptual understandings of the contexts of use. Most importantly, this suggests that social-technical design should primarily be understood as a conceptual enterprise. It is not the empirical accounts of social reality that command design, it is the conceptual

construction of the social-technical imagination that commands an engagement with the contexts of use. Second, this engagement relies on conceptual material, which may involve the cultural models held by the users, or just as importantly, it may be constructed as a local, professional or disciplinary culture on these foundations. Making understandings about the contexts of use available for conceptual construction is a challenge which may not be met by the epistemic strategy of one-off focused research, common in human-centered approaches. My analysis of Sakai has suggested that the socially and temporally distributed epistemic strategies of storing and sharing, supported by the creation of a knowledge base and a community, can mediate the contexts of use in a way that makes them more accessible in the process of conceptual construction.

The social-technical imagination understood along these lines may be similar to the sociological imagination of social theory because of a shared interest in the patternings of human activity, but it is also different in that it applies this interest to the design of artifacts. I have suggested that Actor-network theory (ANT) may be seen as an example of social theory which theorizes the coupling across humans and artifacts in a distributed process. In the next section I will take this insight further to speculate that the patternings of human activities which ANT has theorized as the social phenomenon, may be said to originate (at least to some extent) in the mind, and in the distributed cognitive activity of design.

10.1.4. Reinserting conceptual construction within social theory

Latour (1992) has argued that artifacts participate in the effects that have been theorized as the social. Artifacts constitute the missing mass of social theory, the essential element that would be needed to explain the social phenomenon:

> "According to some physicists, there is not enough mass in the universe to balance the accounts that cosmologists make of it. They are looking everywhere for the 'missing mass' that could add up to the nice expected total. It is the same with sociologists. […]
>
> I expect sociologists to be much more fortunate than cosmologists, because they will soon discover their missing mass. To balance our accounts of society, we simply have to turn our exclusive attention away from humans and look also at nonhumans. […] What our ancestors, the founders of sociology, did a century ago to house the human masses in the fabric of social theory, we should do now to find a place in a new social theory for the nonhuman masses that beg us for understanding." (p. 1.)

In turn, Law (1992) has given an account of the social phenomenon as a process of ordering and patterning, as opposed to structural properties of order and patterns, and suggested that artifacts partake in the creation of this order:

> "the social is nothing other than patterned networks of heterogeneous materials. This is a radical claim because it says that these networks are composed not only of people, but also of machines, animals, texts, money, architectures – any material that you care to mention. So the argument is that the stuff of the social isn't simply human." (p. 380)

The general lines of the ANT-argument are the following: humans and artifacts are pulled together to the effect of becoming an interrelated web (actor-network) capable to span space and time, and to reenter the local arenas of human practice in a relatively stable form. Actor-networks may be experienced as unchangeable, external forces

because of their stability, which they owe to the inclusion of material elements. This capacity does not go back to either the specific capabilities of humans, or the specific capabilities of artifacts, it is the mingling of the two that creates an effect, which is relational. As Law (1992) puts it, ANT understands organizations, devices, representational systems and even human agents as interactive effects arising out of a heterogeneous network of humans and artifacts.

ANT has had a sustained interest in how networks are pulled together. Law has talked about heterogeneous engineering (1987) and Latour has argued that the social has been "assembled" from heterogeneous elements (2005). Latour and Woolgar's (1986) account of immutable mobiles describes how scientific work involves investing the outcomes of research with degrees of facticity. One important aspect of adding facticity to research results is bringing them to a format that can travel across sites – initially between the local sites in the laboratory, and eventually across a geographically dispersed network of scientific institutions. The term immutable mobile refers to the output of this work, which is described as both fixed and capable of travelling, and becoming reinserted into local activities at dispersed sites. Among the immutable mobiles of science we find scientific "facts" and theories, as well as equipment that is produced on the basis of the latter. Latour and Woolgar's analysis suggests that scientific facts are assembled (or socially constructed) in the laboratory by the use of technologies such as writing. It may be suggested that networked software systems are like immutable mobiles since they are assembled in a way that makes them capable to travel and become reinserted at local sites, where they originate local dynamics responsive to a generic logic. In this light infrastructural implosion is a significant source of social patterning.

Actor-networks and their relational effect have been described as the product of the mingling of semiotic and material elements. Latour and Woolgar (1986) have argued that semantic operations such as inscription contribute to the social construction of immutable mobiles. Callon (1986b) has talked about processes of translation which turn material constituents into participants of an actor-network. ANT in general suggests that SCOT's program of separating society and technology is untenable – social interpretation is not an explanation of the technical, interpretation and technology go hand in hand to produce the phenomenon of the social, as the principle of generalized symmetry holds (Callon, 1987; Wyatt, 2008).

At the same time, ANT has not investigated the meaning-making capabilities that make humans capable of "assembling the social" by engaging in the semantic work of translation or inscription with respect to artifacts. Their actors have either been presented as clever strategists, who can make scallops interested (Callon, 1986b), or as academic sleepwalkers, whose actions are commanded by forces beyond their control (Latour & Woolgar, 1986). Meanwhile, ANT has not made clear what humans do in assembling the actor-networks of the social, and has remained wanting an account of human agency. The distributed cognitive framework, and more specifically the distributed cognitive account of social-technical imagination can serve as the missing account of agency within the broader social theoretical framing of Actor-Network theory.

It appears that the human mind is another missing mass in the account of the social phenomenon, which has been overlooked by ANT, but to understand its contribution, a distributed outlook is needed. On the distributed cognition account, design builds on conceptual understandings originating in the contexts of use to envisage new

artifacts through a distributed form of conceptual modeling, which involves cognitive partnership with fragmentary, tentative forms of the artifact. I have suggested that the social technical imagination envisages patternings in human activity with software in such a distributed cognitive process. Thus, the patternings identified by ANT as the social phenomenon may have their origin in the conceptual patterns created by the human mind. It is human cognition, with its characteristic conceptual, pattern-making logic that may be identified at the origin of the assemblies of humans and artifacts, and the patterns may turn out to be at least partially originating in a distributed cognitive process of conceptual modeling. As ANT reminds us, the extent to which the human and material elements can be made "interested" (Callon, 1986b) in the performance of these patternings constitute the bounds of imagination.

10.2. Limitations

The research looked at a single case. While the single case study approach is common in the STS field, a series of similar case studies would allow us to investigate the hypothesis that variations of the procedural logics described in the research are common and typical to an area of software development. It would also allow us to make comparisons and describe the range of situations when the epistemic strategies are found to be applied.

Related to this it should be noted that the Sakai community has been relying on a range of practices and methods in software development to create its own unique, idiosyncratic approach to the making of large-scale software. Because of this, the development processes of Sakai will not be representative or typical of other sites where software is made. The open source movement prides itself on a strong identity of difference with respect to traditional, top-down, managerial approaches to software

development, and Sakai is special within the open source scene in that it has created a community of employees in a particular application domain. There are other communities which have attempted to follow a similar path, as I have pointed out in Chapter 4, but their similarities and differences would require further study. I have further suggested that the embedding of Sakai within a larger reusable software landscape of component software and platform technologies relates its processes of innovation to what we see in case of other web-based platform technologies, such as social media or service-oriented architectures in industry.

I will also add that my goal in the research has not been to describe a typical or desirable software development process, but instead to point out what is *possible* at the outset of the 21st century, in the context of an interconnected, evolving software landscape. My focus has been on the role of human agency in this process, most importantly in creating new epistemic spaces of possibility for future software systems. The general approach has its precedents in microhistory (Brewer, 2010), which grew out of a discontent with mainstream history's focus on universalistic, homogeneous and inexorable processes of modernization. Microhistory is attentive to the local tactics, which outline the contours of human agency and freedom (de Certeau, 1984). In a similar vein, my analysis has been looking at local epistemic strategies and conceptual tactics that contributors deploy for making their influence on software, and I have been attempting to generalize on this level of epistemic practices. While the contexts of making software may represent distinct and divergent configurations, they may all share in the distributed application of general human cognitive capabilities and epistemic strategies.

The research followed a strategy of non-participant observation, without direct engagement with participants. This implies a limitation in terms of the nature of the available materials. Most importantly, local discussions were limited to Sakai participants, and provided practically no information about the institutional mandate of participants. Related to this, I would speculate that the majority of participants had significant freedom in defining their day-to-day strategies, which also included the epistemic approaches to innovation. In the first place, the Sakai community was an organizational experiment, and many institutional members came with a readiness to endorse open-ended innovative exploration to further the domain-driven design of their educational software platform. This was especially true for those who were engaged with Sakai 3. Thus, the process was truly bottom up, in the sense that Sakai participants sought buy-in for the Sakai 3 project from their institution. As I have described in the analysis, the participant from Georgia Tech expressed three distinct institutional priorities at three different moments during the process.

At the same time, we may also speculate that differences in local organizational arrangements had an influence on the different styles of work that could be observed among participants. CARET, the instructional technology department at the University of Cambridge in the UK stood apart from its US counterparts in many respects. Most importantly, CARET could be seen defining itself as a research unit more than any other contributing departments. CARET had for example financed some its Sakai 3 related activities from a research grant, while its deployment of educational technology and related service obligations were more limited than what was typical for US universities. CARET's managers participated directly in the design of Sakai 3, and CARET

participants were active in defining innovative pathways both at the back-end and front-end. Three of the US universities could be further seen to have a distinct profile. The University of Berkeley and the University of Stanford have a profile of active engagement in software related innovation in the educational area, and they had active local design communities. NYU ended up spearheading the completion of the Sakai 3 project, mostly on account of its local institutional agenda that prescribed the choice of a more appropriate new platform. It came to the project supported by an internal coalition of schools, which also brought their own special requirements. The above are only snapshots of differences in local organizational arrangements. Because the observational strategy I adopted did not give me access to these organizational settings, I was not able to study how these local organizational arrangement might have contribute to the epistemic strategies of Sakai 3, and how they might have contributed to the formation of a design space, and the related processes of innovation.

As the development process became redesigned under the managed project, significant portions of the work were pursued in channels that were not part of the web of discourse, and had little online documentation. The research was also missing out on non-documented episodes in earlier phases of development. Based on the available information, my assessment is that the overall outlines of the development could be reliably established from the available materials. Access to more fine-grained discussions would have allowed an opportunity to identify and analyze more local episodes of conceptual construction.

The Sakai OAE platform studied in the research has not yet reached the maturity of a full scale implementation beyond the initial pilots. Because of this, the study was

limited to the design phase, and could not consider a full cycle of design and use. In particular, there was no opportunity to study how the novel logic of the interface the modular, configurational character of the platform would be taken up in the real world contexts of higher education. While a use-based study would have expanded and nuanced the current findings, the grounded-theory analysis of the design process in itself provided rich materials for analysis, and yielded a coherent narrative on its own terms.

10.3. Contributions

The dissertation has engaged with research in distributed cognition, design research, Human-Centered Computing (HCC) and Science and Technology Studies (STS). It has focused in particular on the fields of Social Constructivist Studies of Technology (SCOT), Actor-Network Theory, Social Informatics (SI), Computer-Supported Cooperative Work (CSCW) and distributed cognition in science and engineering. The contributions to these fields are as follows:

- An account of the distributed cognitive processes of innovation in social software for an institutional context.
- An account of the organizational phenomenon of domain-driven (open source) software development.
- An account of the open-ended design space behind innovation in software.
- An account of the possibilizing, expansive character of software design in terms of the generative contribution of human cognition.
- An account of the role of framing devices in providing directionality to distributed cognitive processes of innovation.

- An account of social-technical imagination as a distributed cognitive process which relies on conceptual understanding originating in the contexts of use to envisage patternings in the uses of software.
- An account of infrastructural implosion as the translation of an evolving landscape of software to local processes of innovation in terms of an open-ended design space.

10.4. Implications for further work and practice

Monteiro, Pollock, Hanseth and Williams (2013) have recently argued that human-centered approaches, most notably within CSCW, have failed to engage with the infrastructural character of software, and suggested that taking Information Infrastructures seriously implies a reconceptualization of the role of design. My analysis has shown that a case study approach relying on the distributed cognitive framework can be fruitfully used to investigate the processes of design, providing useful insight about how we may reconceptualize the design process, as suggested by the authors. Understanding the role of an evolving infrastructural landscape and its connections to an epistemic approach relying on conceptual innovation would require further case studies that apply the framework of distributed cognition. The connection of standardization with innovation, the role of platform technologies in fostering local innovation, and within this area, the role of social media platforms and service-oriented architectures are distinct areas which can be expected to yield fruitful contributions in this respect.

Related to the concept of social-technical imagination, my analysis has indicated two threads which could be fruitfully explored in connection with human-centered design practices: engagement with generic infrastructural arrangements for the purpose of

human-centered design, and a distributed strategy that would make understandings about the contexts of use available for design.

It appears that the architectural model of back-end components played a significant role in defining the generic outlines of the system's capabilities for supporting social patternings. At the same time, it appears that these capabilities were not evident with respect to the components, and participants were short of conceptual approaches for making sense of how they would play out in the educational context. Human-centered design has been mostly active in the area of user-facing software, but the social implications of infrastructural arrangements suggest that it should expand its focus to the study and design of infrastructural technologies. I have suggested that human-centered implications of component technologies are currently difficult to grasp for all participants, which would warrant the exploration of conceptual approaches that would help participants with various perspectives in making sense of their capabilities. With respect to designing these components and related systems from a human-centered perspective, I have underlined the potential of framing devices, which would support a practical engagement with how social patternings emerge from the architectural arrangements of back-end technologies. With respect to both of these problems, it appears that human-centered approaches could play a leading role in providing tools and approaches to a broader field of professionals.

Dourish and Bell (2012) outlined a vision where human-centered researchers become the guardians and producers of knowledge about the contexts of use, which they relay to design. My analysis has suggested that related distributed strategies should look beyond the local contribution of research. The implications may be pursued in terms of

knowledge management and professional education. With respect to the former area, my analysis suggested that conceptual understandings embedded in rich cultural contexts can become productive in processes of conceptual construction, and they also provide the foundations for a distributed strategy of knowledge management. I would speculate that the conceptual models that would be central to this endeavor can be understood along the lines described by Shore (1996) in his account of culture as a collection of models. It appears that ethnographies of mobile phone use, some of which came out of iSchools with a human-centered orientation (Katz & Aakhus, 2002; Katz, 2003), have found a research and presentation format, which channels empirically grounded conceptual understandings in connection with mobile sociability without attempting to directly influence the design process.

The problem of education replaces the problem of knowledge in a broader timeframe, and allows us to formulate our question in the following manner: what is it that contributors to human-centered design should ideally know? It appears that the first decades of interactive computing have produced a professional community of human-computer interaction, which is able to navigate the system of user interactions which arises from the coupling of user and software interface. My analysis suggests that there is the possibility for a parallel professional and disciplinary community, who engage with the system emerging from software platforms and patterns of use from a design perspective. Members of such a community would need to have a good footing in both software and the social world. A possible approach would be to provide complementary education to those who have foundations in either one of the fields, and the related

challenge for human-centered computing is to define what this complementary education should involve.

REFERENCES

Abowd, G. D. (2012). What next, ubicomp? Celebrating an intellectual disappearing act. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (pp. 31-40). New York, NY, USA: ACM.

Abowd, G. D. (2014). Ubicomp and Health. GVU Brown Bag lecture recorded at the Georgia Institute of Technology. [streaming video] Retrieved on March 12, 2014 from http://www.gvu.gatech.edu/events/gvu-brown-bag-seminar-gregory-abowd

Abowd, G. D., & Mynatt, E. D. (2000). Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction,* 7(1), 29-58.

Ackerman, M. S. (2000). The intellectual challenge of CSCW: the gap between social requirements and technical feasibility. *Human–Computer Interaction*, *15*(2-3), 179-203.

Agre, P. (1997). Toward a Critical Technical Practice: Lessons Learned in Trying to Reform AI. In Bowker, G., Star, S.L., Turner, W. & Gasser, L. (Eds.), *Social Science, Technical Systems and Cooperative Work: Beyond the Great Divide*. L. Erlbaum Assoc. Inc., Hillsdale, NJ, USA.

Allen, J. P., Rosenbaum, H., & Shachaf, P. (2007). Web 2.0: A Social Informatics View. In: *Proceedings of the Thirteen Americas Conference on Information Systems*, Keystone, CO, USA.

Andersen, H. (2010). Joint Acceptance and Scientific Change: A Case Study. *Episteme* 7(3), 248-265.

Andersen, H., Barker, P. & Chen, X. (2006) *The Cognitive Structure of Scientific Revolutions*. Cambridge, UK: Cambridge UP.

Anderson, R. J. (1994). Representations and requirements: The value of ethnography in system design. *Human-computer interaction*, *9*(3), 151-182.

Androutsellis-Theotokis, S. (2010). Open source software: A survey from 10,000 feet. *Foundations and Trends® in Technology, Information and Operations Management*, *4*(3-4), 187-347.

Atkins, D. E., Brown, J. S., & Hammond, A. L. (2007). *A review of the open educational resources (OER) movement: Achievements, challenges, and new opportunities*. Report to The William and Flora Hewlett Foundation. URL: http://www.hewlett.org/uploads/files/ReviewoftheOERMovement.pdf

Baldwin, C. Y., K. B. Clark (2006). The architecture of participation: Does code architecture mitigate free riding in the open source development model? *Management Science*, 52, 1116–1127.

Bannon, L. J. (2005). A human-centred perspective on interaction design. In: Isomäki, H., Roast, C., & Saariluoma, P. (Eds.). *Future interaction design.* London, UK: Springer. 31-51.

Bannon, L. J. (2010). Approaches to software engineering: a human-centred perspective. In Forbrig, P., Gulliksen, J., & Lárusdóttir, M. K. (Eds.), *Human-Centred Software Engineering - Third International Conference, HCSE 2010, Reykjavik, Iceland, October 14-15, 2010. Proceedings* (pp. 1-5). Berlin, Heidelberg, Germany: Springer.

Bannon, L. J. (2011). Reimagining HCI: toward a more human-centered perspective. *Interactions*, 18(4), 50-57.

Bannon, L. J., & Schmidt, K. (1989). CSCW: Four characters in search of a context. *DAIMI Report Series*, *18*(289).

Bannon, L., & Bødker, S. (1997). Constructing common information spaces. In *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work* (pp. 81-96). Springer Netherlands.

Bardzell, S. (2010). Feminist HCI: taking stock and outlining an agenda for design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1301-1310). New York, NY, USA: ACM.

Bauman, Z., & May, T. (2001). *Thinking sociologically* (2nd ed.). Oxford, UK; Malden, MA, USA: Wiley-Blackwell.

Beaudouin-Lafon, M., & Mackay, W. (2007). Prototyping tools and techniques. In Sears, A., & Jacko, J. A. (Eds.), *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*. Hillsdale, NJ, USA: L. Erlbaum Assoc. Inc.

Becvar, A., Hollan, J., & Hutchins, E. (2008). Representational gestures as cognitive artifacts for developing theories in a scientific laboratory. In Ackerman, M.S., Halverson, C.A., Erickson, Th. & Kellogg, W.A. (Eds.), *Resources, Co-Evolution and Artifacts* (pp. 117-143). London, UK: Springer.

Bell, G. & Dourish, P. (2007). Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. *Personal and Ubiquitous Computing*, 11(2), 133-143.

Belt, H. Van den, and Rip, A. (1987). The Nelson-Winter-Dosi Model and Synthetic Dye Chemistry. In Bijker, W.E., Hughes, T.P. & Pinch, T.J. (Eds.), *The Social Construction of Technological Systems. New Directions in the Sociology and History of Technology* (pp. 187-199). Cambridge, MA, USA: The MIT Press.

Benkler, Y. (2002). Coase's Penguin, or, Linux and "The Nature of the Firm". *Yale Law Journal*, 112(3), 369-446.

Beyer, H. and Holtzblatt, K. 1998. *Contextual Design: Defining Customer-Centered Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Bijker, W. E. (1992). The Social Construction of Fluorescent Lighting, or How an Artefact was Invented in Its Diffusion Stage. In Bijker, W & Law, J (Eds.), *Shaping Technology Building Society*. Cambridge, MA, USA: The MIT Press.

Bijker, W. E. (1993). Do not despair: there is life after constructivism. *Science, Technology & Human Values*, *18*(1), 113-138.

Bijker, W. E. (1997). *Of Bicycles, Bakelites and Bulbs: Toward a Theory of Socio-technical Change*. Cambridge, MA, USA: The MIT Press.

Bijker, W.E. & Law, J. (1992) General Introduction. In: Bijker, W. E., & Law, J. (1992). *Shaping Technology/Building Society: Studies in Socio-technical Change*. Cambridge, MA, USA: The MIT Press.

Bødker, S. (2006). When second wave HCI meets third wave challenges. In *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles* (pp. 1-8). New York, NY, USA: ACM.

Bødker, S., & Grønbæk, K. (1991). Design in action: From prototyping by demonstration to cooperative prototyping. In Greenbaum, J., & Kyng, M. (Eds.). *Design at work: Cooperative design of computer systems* (pp. 197-218). Hillsdale, NJ: USA, Lawrence Erlbaum.

Bollinger, T., Nelson, R., Turnbull, S., & Self, K. (1999). From the editor-response: open-source methods: peering through the clutter. *IEEE Software*, *16*(4), 6-11.

Bonino, M. J., & Spring, M. B. (1991). Standards as change agents in the information technology market. *Computer Standards & Interfaces*, 12(2), 97-107.

Borup, M., Brown, N., Konrad, K., & Van Lente, H. (2006). The sociology of expectations in science and technology. *Technology Analysis & Strategic Management*, 18(3-4), 285-298.

Bourdieu, P. (1977). *Outline of a Theory of Practice*. Cambridge, UK: Cambridge UP.

Bowker, G. C., & Star, S. L. (1998). Building information infrastructures for social worlds – the role of classifications and standards. In Ishida, T. (Ed.). (1998). *Community Computing and Support Systems: Social Interaction in Networked Communities* (pp. 231-248). Berlin, Germany: Springer.

Bowker, G. C., & Star, S. L. (2000). *Sorting Things Out: Classification and Its Consequences*. Cambridge, MA, USA: The MIT Press.

Brewer, J. (2010). Microhistory and the histories of everyday life. *Cultural and Social History*, *7*(1), 87-109.

Bucciarelli, L. L. (1994). *Designing Engineers*. Cambridge, MA, USA: The MIT Press.

Bucher, T. (2012). *Programmed sociality: A software studies perspective on social networking sites*. Doctoral dissertation, University of Oslo.

Bucher, T. (2013) Objects of intense feeling: The case of the Twitter APIs. *Computational Culture* (3).

Button, G. (2000). The ethnographic tradition and design. *Design Studies*, *21*(4), 319-332.

Button, G., & Dourish, P. (1996). Technomethodology: paradoxes and possibilities. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 19-26). New York, NY, USA: ACM.

Callon, M. & B. Latour (1992). Don't Throw the Baby Out with the Bath School! A Reply to Collins and Yearley. In Pickering, A. (Ed.), *Science as Practice and Culture* (pp. 343-368). Chicago, IL, USA: Chicago University Press.

Callon, M. (1980). Struggles and negotiations to define what is problematic and what is not. In Knorr, W.R., Krohn, R. & Whitley, R. P. (Eds.), *The Social Process of Scientific Investigation* (pp. 197-219). Springer Netherlands.

Callon, M. (1986a). The Sociology of an Actor-Network: the Case of the Electric Vehicle. In M. Callon, J. Law and A. Rip (Eds.), *Mapping the Dynamics of Science and Technology: Sociology of Science in the Real World* (pp. 19-34). London, UK: Macmillan.

Callon, M. (1986b). Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of Saint Brieuc Bay. In J. Law (Ed.), *Power, Action and*

Belief: a new Sociology of Knowledge? (pp. 196-233). London, Routledge and Kegan Paul.

Callon, M. (1987). Society in the making: the study of technology as a tool for sociological analysis. In Bijker, W.E., Hughes, T.P. & Pinch, T.J. (Eds.), *The Social Construction of Technological Systems. New Directions in the Sociology and History of Technology* (pp. 83-103), Cambridge, MA, USA: The MIT Press.

Capra, E., Francalanci, C., Merlo, F., & Lamastra, C. R. (2009). A survey on firms' participation in open source community projects. In Boldyreff, C., Crowston, K., Lundell, B. & Wasserman, A.I. (Eds.) *Open Source Ecosystems: Diverse Communities Interacting* (pp. 225-236). Berlin, Heidelberg, Germany: Springer.

Carey, S. (2009). *The Origin of Concepts*. Oxford, UK: Oxford UP.

Charmaz, K. (2006). *Constructing Grounded Theory. A Practical Guide through Qualitative Analysis*. London, UK; Thousand Oaks, CA, USA: Sage Publications.

Charmaz, K. (2010) Grounded Theory as an Emergent Method. In: Hesse-Biber, S. N., & Leavy, P. (Eds.) *Handbook of Emergent Methods*. New York, NY, USA: The Guilford Press.

Cho, J., & Trent, A. (2006). Validity in qualitative research revisited. *Qualitative Research*, 6(3), 319-340.

Christensen, B. T., & Schunn, C. D. (2009). The role and impact of mental simulation in design. *Applied Cognitive Psychology*, *23*(3), 327-344.

Cohen, M. D., March, J. G., & Olsen, J. P. (1972). A garbage can model of organizational choice. *Administrative science quarterly*, 17(1).

Committee on Quality of Health Care in America, Institute of Medicine. (2001). *Crossing the quality chasm: A new health system for the 21st century*. National Academies Press.

Constant, E. W. (1973). A model for technological change applied to the turbojet revolution. *Technology and Culture*, *14*(4), 553-572.

Cooper, A. (2004). *The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity*. Indianapolis, IN, USA: Sams Publishing.

Cooper, A., Reimann, R. & Cronin, D. (2007). *About Face 3: The Essentials of Interaction Design*. Indianapolis, IN, USA: Wiley Publishing.

Corbin, J. M. & Strauss, A. (1990). Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, 13(1), 3-21.

Corbin, J. M. & Strauss, A. (2007). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (3rd ed.). Los Angeles, CA, USA: Sage Publications.

Creswell, J. W. & Miller, D. L. (2000). Determining validity in qualitative inquiry. *Theory into Practice*, 39(3), 124-130.

Cross, N. (2000). *Engineering Design Methods: Strategies for Product Design*. Chichester, UK: Wiley.

Crowston, K., Wei, K., Howison, J., & Wiggins, A. (2012). Free/Libre open-source software development: What we know and what we do not know. *ACM Computing Surveys*, *44*(2), 7.

De Certeau, M. (1984). *The Practice of Everyday Life*. Berkeley, CA, USA: University of California Press.

DeMillo, R. A. (2011). *Abelard to Apple: The Fate of American Colleges and Universities.* Cambridge, MA, USA: The MIT Press.

Dibona, C., Stone, M., & Ockman, S. (1999). *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA: O'Reilly & Associates.

Donmoyer, R. (2001). Paradigm Talk Reconsidered. In Richardson, V. (Ed.), *Handbook of Research on Teaching* (4th ed., pp. 174–97). Washington, DC, USA: American Educational Research Association.

Dorst, K., & Cross, N. (2001). Creativity in the design process: co-evolution of problem–solution. *Design Studies*, *22*(5), 425-437.

Dosi, G. (1982). Technological paradigms and technological trajectories: a suggested interpretation of the determinants and directions of technical change. *Research Policy*, *11*(3), 147-162.

Dourish, P. & Bell, G. (2011). *Divining a Digital Future: Mess and Mythology in Ubiquitous Computing*. Cambridge, MA, USA: The MIT Press.

Dourish, P. (2006). Implications for design. *Proceedings of the SIGCHI conference on Human Factors in computing systems* (pp. 541-550). New York, NY, USA: ACM.

Dourish, P., & Bell, G. (2008). "Resistance is futile": reading science fiction alongside ubiquitous computing. *Personal and Ubiquitous Computing*, 11(2), 133-143.

Du Gay, P. (1997). *Doing cultural studies: The story of the Sony Walkman.* London, UK; Thousand Oaks, CA, USA: Sage Publications.

Edwards, W. K., & Grinter, R. E. (2001). At home with ubiquitous computing: seven challenges. In *Ubicomp 2001: Ubiquitous Computing* (pp. 256-272). Berlin, Heidelberg, Germany: Springer.

Egyedi, T. M., & Mehos, D. C. (Eds.). (2010). *Inverse Infrastructures: Disrupting Networks from Below*. Cheltenham, UK; Northhampton, MA, USA: Edward Elgar Publishing.

Egyedi, T. M., Vrancken, J. L., & Ubacht, J. (2007). Inverse infrastructures: coordination in self-organizing systems. In *5th International Conference on Standardization and Innovation in Information Technology* (pp. 23-36). IEEE.

Elliott, M. S. & Kraemer K. L. (2012a). *Computerization Movements and Technology Diffusion. From Mainframes to Ubiquitous Computing*. Medford, NJ, USA: Information Today, Inc.

Elliott, M. S. & Kraemer K. L. (2012b). Comparative Perspective on Computerization Movements: Implications for Ubiquitous Computing. In Elliott, M. S. & Kraemer K. L.

(2012) *Computerization Movements and Technology Diffusion. From Mainframes to Ubiquitous Computing*. Medford, NJ, USA: Information Today, Inc.

Erickson, T. (2013): Social Computing. In Soegaard, M. & Dam, R. F. (Eds.). *The Encyclopedia of Human-Computer Interaction* (2nd edition). Aarhus, Denmark: The Interaction Design Foundation. Available online at http://www.interaction-design.org/encyclopedia/social_computing.html

Evans, D. D. S., Hagiu, A., & Schmalensee, R. L. (2006). *Invisible Engines*. Cambridge, MA, USA: The MIT Press.

Feller, J., & Fitzgerald, B. (2002). *Understanding Open Source Software Development*. London, UK: Addison-Wesley.

Fielding, R. T. (1999). Shared leadership in the Apache project. *Communications of the ACM*, *42*(4), 42-43.

Fielding, R. T. (2005). Software architecture in an open source world. In *Proceedings. 27th International Conference on Software Engineering* (p. 43). IEEE.

Fitzgerald, B. (2006). The transformation of open source software. *MIS Quarterly*, 587-598.

Fleck, J. (1988). Innofusion or diffusation? The nature of technological development in robotics. Research Centre for Social Sciences, University of Edinburgh.

Fleck, J. (1993). Configurations: crystallizing contingency. *International Journal of Human Factors in Manufacturing*, *3*(1), 15-36.

Fleck, J. (1995). Configurations and standardization. In Esser, J. Fleischmann, G. & Heimer, T. (Eds.), *Soziale und Okonomische Konflikte in Standardisierungsprozessen* (pp. 38-65). Frankfurt, Germany: Campus Verlag.

Folkmann, M. N. (2013). *Aesthetics of Imagination in Design*. Cambridge, MA, USA: MIT Press.

Foucault, M. (1976). *The Archaeology of Knowledge*. New York, NY, USA: Harper and Row.

Foucault, M. (1980). *Power/Knowledge.* New York, NY, USA: Pantheon.

Gacek, C., & Arief, B. (2004). The many meanings of open source. *Software, IEEE*, *21*(1), 34-40.

Garcia, A.C., Standlee, A.I., Bechkoff, J., Cui, Y. (2009). Ethnographic Approaches to the Internet and Computer-Mediated Communication. *Journal of Contemporary Ethnography*, 38 (1), 52-84.

Gawer, A., & Cusumano, M. A. (2002). *Platform Leadership*. Boston, MA, USA: Harvard Business School Press.

Gero, J. S. (1990). Design prototypes: a knowledge representation schema for design. *AI magazine*, 11(4), 26-36.

Giddens, A. (1979). *Central Problems in Social Theory: Action, structure, and contradiction in social analysis.* Berkeley, CA, USA: University of California Press.

Giddens, A. (1984). *The Constitution of Society: Outline of the Theory of Structuration*. Berkeley, CA, USA: University of California Press.

Giddens, A. (2009). *Sociology* (6th ed.). Cambridge, UK; Malden, MA, USA: Polity Press.

Glaser, B. G. (2001). *The Grounded Theory Perspective: Conceptualization Contrasted with Description*. Mill Valley, CA, USA: Sociology Press.

Glaser, B. G., & Strauss, A. L. (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicago, IL, USA: Aldine de Gruyter.

Godfrey, M. W., & Tu, Q. (2000). Evolution in open source software: A case study. In *Proceedings of the International Conference on Software Maintenance* (pp. 131-142). Washington, DC, USA: IEEE Computer Society.

Goffman, E. (1974). *Frame Analysis: An Essay on the Organization of Experience*. New York, NY, USA: Harper & Row.

Goldschmidt & Badke-Schaub (2010). The Design-Psychology Indispensible Research Partnership. In Dorst, K., Stewart, S., Staudinger, I., Paton, B. & Dong, A. (Eds.), *Proceedings of the 8th Design Thinking Research Symposium* (pp. 199-209).

Goodwin, C. (1995). Seeing in depth. *Social studies of science*, *25*(2), 237-274.

Grinter, R. E., Edwards, W. K., Newman, M. W., & Ducheneaut, N. (2005). The work to make a home network work. In *Proceedings of the 2005 Ninth European Conference on Computer-Supported Cooperative Work*. Springer Netherlands. pp. 469-488.

Grudin, J. (1991). CSCW: The convergence of two development paradigms. *Proceedings of the ACM CHI'91 Human Factors in Computing Systems Conference* (pp. 91-97). New York, NY, USA: ACM.

Grudin, J. (1992). Utility and usability: research issues and development contexts. *Interacting with Computers*, *4*(2), 209-217.

Grudin, J. (2012) Introduction: A moving target—The evolution of human–computer interaction. In Jacko, J. (Ed), *Human-Computer Interaction Handbook: Fundamentals, evolving technologies, and emerging applications*. CRC Press.

Hall, R., Wieckert, K., & Wright, K. (2010). How does cognition get distributed? Case studies of making concepts general in technical and scientific work. In Banich, M. T., & Caccamise, D. (Eds.), *Generalization of knowledge: Multidisciplinary perspectives* (pp. 225-246). New York, NY, USA: Psychology Press.

Hanganu, G. (2008). *The community source development model*. OSSWatch. Retrieved form URL: http://www.oss-watch.ac.uk/resources/communitysource

Hanseth, O., & Lyytinen, K. (2010). Design theory for dynamic complexity in information infrastructures: the case of building internet. *Journal of Information Technology*, *25*(1), 1-19.

Hanseth, O., Monteiro, E., Hatling, M. (1996). Developing Information Infrastructure: The Tension between Standardization and Flexibility. *Science, Technology, & Human Values*, 21(4), 407-426.

Harmon, E. & Nersessian, N. J. (2008). Cognitive partnerships on the bench top: Designing to support scientific researchers. In *Proceedings of the 7th ACM conference on Designing interactive systems* (pp. 119-128). New York, NY, USA: ACM.

Harrison, S., Tatar, D. & Sengers, P. (2007). The three paradigms of HCI. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM.

Hars, A. & Ou, S. S. (2002). Working for free? Motivations for participating in open-source projects. *International Journal of Electronic Commerce*. 6(3), 25-39.

Henderson, K. (1991). Flexible sketches and inflexible data bases: Visual communication, conscription devices, and boundary objects in design engineering. *Science, Technology & Human Values*, *16*(4), 448-473.

Hine, C. (2000). *Virtual Ethnography*. London, UK; Thousand Oaks, CA, USA: Sage Publications.

Hollan, J., Hutchins, E., Kirsh, D. (2000). Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction*, 7(2), 174-196.

Hughes, T. P. (1986). The seamless web: technology, science, etcetera, etcetera. *Social Studies of Science*, *16*(2), 281-292.

Hughes, T. P. (1987). The evolution of large technological systems. In Bijker, W., Hughes, T.P., Pinch, T. (Eds.), *The social construction of technological systems: New directions in the sociology and history of technology* (pp. 51-82). Cambridge, MA, USA: The MIT Press.

Hughes, T. P. (1993). *Networks of Power: Electrification in Western Society, 1880-1930*. Baltimore, MD, USA: Johns Hopkins University Press.

Hughes, T. P. (1994). Technological momentum. In Smith, M. R., & Marx, L. (Eds.). (1994). *Does Technology Drive History? The Dilemma of Technological Determinism*. Cambridge, MA, USA: The MIT Press.

Hughes, T. P. (2005). From firm to networked systems. *Business History Review*, *79*(3), 587-593.

Hughes, T. P. (2011). *Rescuing Prometheus: Four Monumental Projects That Changed Our World*. New York, NY, USA: Vintage.

Hutchins, E. (1995a). *Cognition in the Wild*. Cambridge, MA, USA: The MIT Press

Hutchins, E. (1995b). How a cockpit remembers its speeds. *Cognitive Science*, *19*(3), 265-288.

Hutchins, E. (1996). Learning to navigate. In Chaiklin, S., & Lave, J. (Eds). *Understanding practice: Perspectives on activity and context*. Cambridge, UK: Cambridge UP.

Hutchins, E. (1999). Cognitive artifacts. *The MIT Encyclopedia of the Cognitive Sciences* (pp 126-127). Cambridge, MA, USA: The MIT Press.

Jaimes, A., Sebe, N., & Gatica-Perez, D. (2006). Human-centered computing: a multimedia perspective. In *Proceedings of the 14th annual ACM international conference on Multimedia (*pp. 855-864*)*. New York, NY, USA: ACM.

Jørgensen, N. (2001). Putting it all in the trunk: incremental software development in the FreeBSD open source project. *Information Systems Journal*, *11*(4), 321-336.

Katz, J. E. (Ed.). (2003). *Machines That Become Us: The Social Context of Personal Communication Technology*. New Brunswick, NJ, USA: Transaction Publishers.

Katz, J. E., & Aakhus, M. (Eds.). (2002). *Perpetual Contact: Mobile Communication, Private Talk, Public Performance*. Cambridge, UK: Cambridge UP.

Kent, W. (1978). *Data and reality: Basic assumptions in data processing reconsidered*. New York, NY, USA: Elsevier Science Inc.

Kirk, J., & Miller, M. L. (1988). *Reliability and Validity in Qualitative Research*. Beverly Hills, CA: Sage Publications.

Kirsh, D. (2009) Interaction, External Representation and Sense Making. In Taatgen, N., van Rijn, H., Schomaker, L. (Eds.) *Proceedings of the Thirty-First Annual Conference of the Cognitive Science Society* (pp. 1103-1108). Mahwah, NJ: Lawrence Erlbaum.

Kline, R., & Pinch, T. (1996). Users as agents of technological change: The social construction of the automobile in the rural United States. *Technology and Culture*, *37*(4), 763-795.

Kling, R. & Iacono, S. (1988). The mobilization of support for computerization: The role of computerization movements. *Social Problems*, 35(3), 226-243.

Kling, R. & Iacono, S. (2001). Computerization Movements: The Rise of the Internet and Distant Forms of Work. In: Yates, J. & Van Maanene, J. (Eds.) *Information Technology and Organizational Transformation: History, Rhetoric, and Practice* (pp. 93-136). Thousand Oaks, CA, USA: Sage Publications. Reprinted in: Elliott&Kraemer (2012b)

Kling, R. & Scacchi, W. (1982). The web of computing: Computer technology as social organization. *Advances in Computers, 21.*

Kling, R. & Star, S. L. (1998). Human centered systems in the perspective of organizational and social informatics. *ACM SIGCAS Computers and Society*, *28*(1), 22-29.

Kling, R. (1980). Social analyses of computing: Theoretical perspectives in recent empirical research. *ACM Computing Surveys*, *12*(1), 61-110.

Kling, R. (1991). Computerization and social transformations. *Science, Technology & Human Values*, *16*(3), 342-367.

Kling, R. (2007). What is social informatics and why does it matter? *The Information Society*, *23*(4), 205-220.

Knorr-Cetina, K. (1981). *The Manufacture of Knowledge*. Oxford, UK: Pergamon Press.

Knorr-Cetina, K. (1984). *The Fabrication of Facts: Toward a Microsociology of Scientific Knowledge*. Konstanz, Germany: Bibliothek der Universität Konstanz.

Knorr-Cetina, K. (1999). *Epistemic cultures: How the sciences make knowledge*. Cambridge, MA, USA: Harvard University Press.

Knorr-Cetina, K. (2004). How are global markets global? The architecture of a flow world. In Knorr-Cetina, K. & Preda, A. (Eds.). *The sociology of financial markets* (pp. 38-61). Oxford, UK: Oxford University Press.

Krishnamurthy, S. (2003) A managerial overview of Open Source software. *Business Horizons* 46(5), 47–56.

Kruger, C., & Cross, N. (2006). Solution driven versus problem driven design: strategies and outcomes. *Design Studies*, 27(5), 527-548.

Kuhn, T. (1970). *The Structure of Scientific Revolutions*. University of Chicago Press.

Latour, B. (1996). *Aramis, or, the Love of Technology.* Cambridge, MA, USA: Harvard University Press.

Latour, B. (2005). *Reassembling the Social. An Introduction to Actor-Network-Theory*. Oxford, UK: Oxford University Press.

Lave, J. (1988). *Cognition in practice: Mind, mathematics and culture in everyday life.* Cambridge, UK: Cambridge U niversity Press.

Law, J. & Callon, M. (1992): The Life and Death of an Aircraft: A Network Analysis of Technical Change. In: Bijker, W. E., & Law, J. (Eds.). *Shaping Technology/Building Society: studies in socio-technical change*. Cambridge, MA, USA: The MIT Press.

Law, J. (1987). Technology and heterogeneous engineering: the case of Portuguese expansion. In Bijker, W. E., Hughes, T. P., & Pinch, T. J. (1987). *The social construction of technological systems: new directions in the sociology and history of technology* (pp. 1-134). Cambridge, MA, USA: The MIT Press.

Law, J. (1992). Notes on the theory of the actor-network: Ordering, strategy, and heterogeneity. *Systems Practice*, 5(4), 379-393.

Lessig, L. (1999). *Code: And Other Laws of Cyberspace*. New York, NJ, USA: Basic Books.

Liikkanen, L., Laakso, L. and Björklund, T. (2011). Foundations for studying creative design practices. In *Procedings of the Second Conference on Creativity and Innovation in Design* (pp. 309-315). New York, NY, USA: ACM.

Liu, Z., Nersessian, N. J., & Stasko, J. (2008). Distributed cognition as a theoretical framework for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6), 1173-1180.

Lovink, G. (2013) A World Beyond Facebook: Introduction to the Unlike Us Reader. In Lovink, G. & Rasch, M. (Eds.) *Unlike US Reader. Social Media Monopolies and Their Alternatives*. Amsterdam, Netherlands: Institute of Network Cultures.

Löwgren, J. (1995). *Perspectives on usability*. IDA Technical Report, Linköping University, Department of Computer and Information Science. Retrieved from URL https://www.academia.edu/2741542/Perspectives_on_usability on Jan. 25 2014.

MacCormack, A., Rusnak, J. & Baldwin, C. Y. (2006). Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science*, 52(7), 1015-1030.

Mayntz, R. & Hughes, T. P. (Eds.) (1988). *The Development of Large Technical Systems*. Frankfurt am Main, Germany: Campus Verlag.

Miles, M., & Huberman, A. (1994). *Qualitative data analysis*. Thousand Oaks, CA, USA: Sage Publications.

Milev, R., Muegge, S., & Weiss, M. (2009). Design evolution of an open source project using an improved modularity metric. In Crowston, C. B. K., & Wasserman, B. L. (Eds.) *Open Source Ecosystems: Diverse Communities Interacting* (pp. 20-33). Springer Berlin Heidelberg.

Miller, G. A., Galanter, E., & Pribram, K. H. (1986). *Plans and the Structure of Behavior*. New York, NY, USA: Adams Bannister Cox.

Mills, C. W. (2000). *The Sociological Imagination*. Oxford, UK: Oxford University Press.

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, *11*(3), 309-346.

Monteiro, E., Pollock, N., Hanseth, O., & Williams, R. (2013). From artefacts to infrastructures. *Computer Supported Cooperative Work*, 22(4-6), 575-607.

Moody, G. (2001). *Rebel Code: Linux and the Open Source Revolution*. New York, NY, USA: Perseus Publishing.

Morse, J. M., Barrett, M., Mayan, M., Olson, K., & Spiers, J. (2008). Verification strategies for establishing reliability and validity in qualitative research. *International Journal of Qualitative Methods*, 1(2), 13-22.

Muller, M.J. (2002). Participatory design: the third space in HCI. In Jacko, J. A. and Sears, A. (Eds.). *The human-computer interaction handbook* (pp. 1051-1068). Hillsdale, NJ: USA, L. Erlbaum Associates Inc.

Nardi, B. A., & Miller, J. R. (1991). Twinkling lights and nested loops: distributed problem solving and spreadsheet development. *International Journal of Man-Machine Studies*, *34*(2), pp. 161-184.

Narduzzo, A., Rossi, A. (2004). The role of modularity in free/open source software development. In Koch, S. (Ed.), *Free/Open Source Software Development* (pp. 84-102). Hershey, PA, USA: Idea Group Publishing.

Nersessian, N. J. (1992a). How do scientists think? Capturing the dynamics of conceptual change in science. In Giere, R. N. (Ed.), *Cognitive Models of Science* (pp. 3-44.). Minneapolis, MN, USA: University of Minnesota Press.

Nersessian, N. J. (1992b) In the Theoretician's Laboratory: Thought Experimenting as Mental Modeling. *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*, Vol. 2. 291-301.

Nersessian, N. J. (2002a). The cognitive basis of model-based reasoning in science. In Carruthers, P., Stich, S. & Siegal, M. (Eds.), *The Cognitive Basis of Science* (pp. 133-153). Cambridge, UK: Cambridge University Press.

Nersessian, N. J. (2002b). Kuhn, conceptual change, and cognitive science. In T. Nichols (Ed.). *Thomas Kuhn*. *Contemporary Philosophers in Focus* Series (pp. 178-211), Cambridge, UK: Cambridge UP.

Nersessian, N. J. (2005). Interpreting scientific and engineering practices: Integrating the cognitive, social, and cultural dimensions. In M. Gorman, R. Tweney, D. Gooding, & A. Kincannon (Eds.) *Scientific and Technological Thinking* (pp. 17-56). Erlbaum.

Nersessian, N. J. (2006). The Cognitive-Cultural Systems of the Research Laboratory. *Organization Studies*, 27(1), 125-145.

Nersessian, N. J. (2008b). Mental Modeling in Conceptual Change. In Vosniadou, S. (Ed.), *International Handbook of Research on Conceptual Change* (pp. 391-416). London, UK: Routledge.

Nersessian, N. J. (2012). Engineering Concepts: The Interplay between Concept Formation and Modeling Practices in Bioengineering Sciences. *Mind, Culture, and Activity*, 19(3), 222-239.

Nersessian, N. J., Kurz-Milcke, E., Newstetter, W. C., & Davies, J. (2003). Research laboratories as evolving distributed cognitive systems. In Alterman, R. & Kirsch, D. (Eds.), *Proceedings of the 25th Annual Conference of the Cognitive Science Society*. pp. 857-862. Mahwah, NJ, USA; London, UK: Lawrence Erlbaum.

Nersessian, N.J. (2008a) *Creating Scientific Concepts*. Cambridge, MA, USA: The MIT Press.

Nersessian, N.J. (2009). How do engineering scientists think? Model-based simulation in biomedical engineering laboratories. *Topics in Cognitive Science* 1(4), 730-757.

Nichols, D. M., & Twidale, M. B. (2003). The usability of open source software. *First Monday*, *8*(1-6).

Nielsen, J. (1993). *Usability Engineering*. Boston, MA, USA: AP Professional.

Noll, J. (no date). Innovation in Open Source Software Development. Unpublished manuscript. Retrieved on March 12, 2014 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.158.3785&rep=rep1&type=pdf

Norman, D. A. (1988). *The Psychology of Everyday Things*. New York, NY, USA: Basic Books.

Norman, D. A. (1991). Cognitive artifacts. In Carroll, J.M. (Ed.), *Designing Interaction: Psychology at the Human-Computer Interface*. New York, NY, USA: Cambridge University Press.

Norman, D. A. (1993). *Things that Make Us Smart*. Reading, MA, USA: Addison-Wesley.

Norman, D. A. (2002). *The Design of Everyday Things*. New York, NY, USA: Basic Books.

O'Reilly, T. (1999). Lessons from open source software development. *Communications of the ACM*, *42*(4), 33-37.

O'Reilly, T. (2005). What Is Web 2.0. Retrieved on March 12, 2014 from http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

Orlikowski, W. J. (1992). The duality of technology: Rethinking the concept of technology in organizations. *Organization Science*, *3*(3), 398-427.

Orlikowski, W. J. (1993). Learning from notes: Organizational issues in groupware implementation. *The Information Society*, 9(3). 237–250.

Orlikowski, W. J., & Gash, D. C. (1994). Technological frames: making sense of information technology in organizations. *ACM Transactions on Information Systems* 12(2), 174-207.

Orlikowski, W.J. (2000). Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations. *Organization Science* 11(4), 404-428.

Osbeck, L. M., & Nersessian, N. J. (2006). The distribution of representation. *Journal for the Theory of Social Behaviour*, 36(2), 141-160.

Pahl, G., Beitz, W., Feldhusen, J., Grote, K. H. (1984) *Engineering Design. A Systematic Approach*. London, UK: Springer Verlag.

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, *15*(12), 1053-1058.

Perry, M. (2003) Distributed Cognition. In Carroll, J. M. (Ed.), *HCI Models Theories and Frameworks - Toward Multidisciplinary Science* (pp. 193-224). Burlington, MA, U: Morgan Kaufmann.

Petre, M., & Green, T. R. (1992). Requirements of graphical notations for professional users: electronics CAD systems as a case study. *Le Travail Humain*, 55. 47-70.

Pinch, T. J. and Bijker, W. E. (1987). The social construction of facts and artifacts: or how the sociology of science and the sociology of technology might benefit each other. In Bijker, W. E., Hughes, T. P., & Pinch, T. J. (Eds.). *The social construction of technological systems: new directions in the sociology and history of technology*. Cambridge, MA, USA: The MIT Press.

PratiScienS (2007). Présentation du groupe PratiScienS. Retrieved from http://poincare.univ-nancy2.fr/digitalAssets/98610_Texte-Site-Web-PratiScienS-FINAL-Membres.pdf

Raymond, E.S. (1999). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Cambridge, MA, USA: O'Reilly & Associates.

Reis, C. R., & de Mattos Fortes, R. P. (2002). An overview of the software engineering process and tools in the Mozilla project. In *Proceedings of the Open Source Software Development Workshop* (pp. 155-175).

Ritzer, G. (1975). *Sociology: A Multiple Paradigm Science*. Boston, MA, USA: Allyn and Bacon.

Rogers, Y. (1997). Reconfiguring the social scientist: Shifting from telling designers what to do to getting more involved. In Bowker, G., Star, S.L., Turner, W. & Gasser, L. (Eds.), *Social Science, Technical Systems and Cooperative Work: Beyond the Great Divide* (pp. 57-77). Hillsdale, NJ, USA: Lawrence Erlbaum.

Rogers, Y. (2006). Moving on from Weiser's vision of calm computing: Engaging Ubicomp experiences. In *UbiComp 2006: Ubiquitous Computing* (pp. 404-421). Springer Berlin-Heidelberg.

Römer, A., Pache, M., Weisshahn, G., Lindemann, U., & Hacker, W. (2001). Effort-saving product representations in design – results of a questionnaire survey. *Design Studies*, 22(6), 473-491.

Römer, A., Pache, M., Weißhahn, G., Lindemann, U., & Hacker, W. (2001). Effort-saving product representations in design—results of a questionnaire survey. *Design Studies*, 22(6), 473-491.

Rosenbaum, H. (2008). Web 2.0, 3.0: After the thrill is gone. Invited closing keynote address given at the Faculty Summer institute, University of Illinois, Champaign-Urbana, May 15.

Salen, K., & Zimmerman, E. (2004). *Rules of Play: Game Design Fundamentals*. Cambridge, MA, USA: The MIT Press.

Sawyer, S., Rosenbaum, H. (2000). Social informatics in the information sciences: Current activities and emerging directions. *Informing Science*, 3(2), 89-89.

Scacchi, W. (2002). Understanding the requirements for developing open source software systems. In *Software, IEE Proceedings.* 149(1), 24-39.

Scacchi, W. (2004). Free/Open Source Software Development Practices in the Computer Game Community. *IEEE Software* 21(1), 56-66.

Scacchi, W. (2009). Understanding the requirements for open source software. In Lyytinen, K., Loucopoulos, P., Mylopoulos, J. & Robinson, B. (Eds.), *Design Requirements Engineering: A Ten-Year Perspective* (pp. 467-494). Springer Berlin Heidelberg.

Schank, R.C. & Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding*. Hillsdale, NJ, USA: Earlbaum Assoc.

Schatzki, T. R. (2000) Introduction: Practice Theory. In: Knorr-Cetina, K., Schatzki, T. R., & von Savigny, E. (Eds.), *The Practice Turn in Contemporary Theory*. London, UK: Routledge.

Schmidt, K., & Bannon, L. (1992). Taking CSCW seriously. *Computer Supported Cooperative Work*, 1(1-2), 7-40.

Schön, D. A. (1999). *The Reflective Practitioner*. New York, NY, USA: Basic Books.

Schumpeter, J.S. (1939), *Business Cycles*. New York, NY, USA: McGraw-Hill.

Searle, J. R. (1976). A classification of illocutionary acts. Language in society, 5(01), 1-23.

Sebe, N. (2010) Human-centered Computing. In Nakashima, H., Aghajan, H. K., & Augusto, J. C. (Eds.), *Handbook of ambient intelligence and smart environments* (pp. 349-370). Springer.

Severance, C. (2011) *Sakai: Free as in Freedom (Alpha): A Retrospective Diary*. US: CreateSpace.

Shore, B. (1996). *Culture in Mind: Cognition, Culture, and the Problems of Meaning*. Oxford, UK: Oxford University Press.

Silverman, D. (2006) *Interpreting Qualitative Data*. (3rd ed.) London, UK: Sage Publications.

Simon, H. A. (1996). *The Sciences of the Artificial*. Cambridge, MA, USA: The MIT Press.

Snow, D. A., & Benford, R. D. (1992). Master frames and cycles of protest. In A. D. Morris & C. M. Mueller (Eds.) *Frontiers in Social Movement Theory* (pp. 133-155). New Haven, CT, USA: Yale University Press.

Sommerville, I., Rodden, T., Sawyer, P., Bentley, R., & Twidale, M. (1993). Integrating ethnography into the requirements engineering process. In *Proceedings of IEEE International Symposium on Requirements Engineering, 1993* (pp. 165-173). IEEE.

Star, S. L. (1999). The ethnography of infrastructure. *American Behavioral Scientist*, 43(3), 377-391.

Star, S. L., & Griesemer, J. R. (1989). Institutional ecology, translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science*, 19(3), 387-420.

Star, S. L., & Ruhleder, K. (1996). Steps toward an ecology of infrastructure: Design and access for large information spaces. *Information Systems Research*, 7(1), 111-134.

Stewart, J., & Williams, R. (2005). The wrong trousers? Beyond the design fallacy: social learning and the user. In: Howcroft, D., & Trauth, E. M. (Eds.). *Handbook of Critical Information Systems Research: Theory and Application.* Cheltenham, UK; Northhampton, MA, USA: Edward Elgar Publishing.

Suchman, L. (1993). Do categories have politics? *Computer Supported Cooperative Work*, 2(3), 177-190.

Suchman, L. (1995). Making work visible. *Communications of the ACM,* 38(9), 56-ff.

Suchman, L. (2002). Located accountabilities in technology production. *Scandinavian Journal of Information Systems*, 14(2), 91-106.

Suchman, L. A. (1983). Office procedure as practical action: models of work and system design. *ACM Transactions on Information Systems*, *1*(4), 320-328.

Suchman, L. A. (1987). *Plans and Situated Actions: the Problem of Human-Machine Communication*. Cambridge, UK: Cambridge UP.

Szyperski, C., Gruntz, D., & Murer, S. (2002). *Component Software: Beyond Object-Oriented Programming*. Boston, MA, USA: Addison-Wesley.

Timmermans, S., & Berg, M. (1997). Standardization in action: achieving local universality through medical protocols. *Social studies of science*, *27*(2), 273-305.

Timmermans, S., & Berg, M. (Eds.). (2003). *The Gold Standard: The Challenge of Evidence-based Medicine and Standardization in Health Care*. Temple University Press.

Torvalds, L. (1999). The Linux edge. In DiBona, C., Ockman, S. & Stone, M. (Eds.), *Open Sources: Voices from the Open Source Revolution*. Beijing: O'Reilly.

Trickett, S. B., & Trafton, J. G. (2007). "What if…": The use of conceptual simulations in scientific reasoning. *Cognitive Science*, *31*(5), 843-875.

Tweney, R. R. (1989). A framework for the cognitive psychology of science. In Gholson, B. Houts, A. Neimeyer, R. A. & Shadish, W. (Eds.), *Psychology of Science and Metascience* (pp. 342-366). Cambridge, UK: Cambridge UP.

Valsiner, J. (1987). *Culture and the Development of Children's Action: Cultural-historical Theory of Developmental Psychology*. Chichester, UK; New York, NY, USA: John Wiley & Sons.

Van Lamsweerde, A. (2009). *Requirements engineering: from system goals to UML models to software specifications*. Chichester, UK; New York, NY, USA: John Wiley & Sons.

Van Lente, H. & Rip, A. (1998a) Expectations in technological developments: an example of prospective structures to be filled by agency. In: Disco, C. & van der Meulen, B. (Eds), *Getting New Technologies Together. Studies in Making Socio-technical Order*. Berlin, Germany: De Gruyter.

Van Lente, H. (1993). *Promising technology: the dynamics of expectations in technological developments*. Doctoral dissertation, Universiteit Twente.

Van Lente, H., & Rip, A. (1998b). The Rise of Membrane Technology: From Rhetorics to Social Reality. *Social Studies of Science*, *28*(2), 221-254.

Vertesi, J. (2012) "Seeing Like a Rover": Visualization, Embodiment and Interaction on the Mars Exploration Rover Mission, *Social Studies of Science* 42 (3), pp. 393-414.

Visser, W. (2006) *The cognitive artifacts of designing*. Lawrence Erlbaum.

Visser, W. (2007). Conception individuelle et collective. Approche de l'ergonomie cognitive. In M. Borillo & J.-P. Goulette (Eds.) (2002) *Cognition et création. Explorations cognitives des processus de conception*. Bruxelles, Belgique : Mardaga. pp. 311-327.

Vygotsky, L. L. S. (1978). *Mind in society: The development of higher psychological processes*. Harvard UP.

Wasserman, A.I., Capra, E. (2007). Evaluating software engineering processes in commercial and community Open Source projects. In: *Proc. Int'l. Workshop Emerging Trends in FLOSS Research and Development.*

Wasserman, A.I., Capra, E. (2008). A framework for evaluating managerial styles in Open Source projects. *Proceedings of Open Source Systems Conference*, pp. 1–14.

Weber, M. 1904/1949. Objectivity in Social Science and Social Policy. In E. A. Shils and H. A. Finch (Eds.) *The Methodology of the Social Sciences*, New York, NY, USA: Free Press.

Wheeler, B. (2010). Open Source 2010: Reflections on 2007. *Educause Review*, *42*(1), 49-52.

Williams, R., Edge, D. (1996). The Social Shaping of Technology. *Research Policy* Vol. 25, pp. 856-899.

Winograd, T., & Flores, F. (1987). *Understanding computers and cognition: A new foundation for design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Wittgenstein, L. (1953). *Philosophical Investigations*. Oxford: Macmillan.

Wyatt, S. (2008). Technological determinism is dead; long live technological determinism. In Hackett, E. J., Amsterdamska, O., Lynch, M., & Wajcman, J. (Eds.) *The handbook of science and technology studies* (pp. 165-180). Cambridge, MA, USA: The MIT Press.

Yu, L., Chen, K. (2004). Categorization of common coupling and its application to the maintainability of the Linux kernel. *IEEE Transactions of Software Engineering*. 30(10), 694–706.

Endnotes

[1] Clay Fenlason's blogpost after becoming Product Manager, July 29, 2009, http://sakaipm.wordpress.com/page/3/

[2] https://confluence.sakaiproject.org/display/SAKDEV/Authoring+Summit+-+Use+Cases

[3] http://opensource.org/licenses/ecl2.php

[4] http://www.apache.org/licenses/LICENSE-2.0

[5]See Moodle's guidelines for code contribution, http://docs.moodle.org/dev/Guidelines_for_contributed_code#The_Add-on_Frequently_Asked_Question_.28FAQ.29

[6] See the Apache Software Foundation's documentation for Incubation Projects at https://incubator.apache.org/incubation/Incubation_Policy.html

[7] Architectural information about S/CLE relies on the following document, unless otherwise noted: Sakai Java Framework. Version 1.0. Technical Report, Sakai Project. December 3, 2004. Prepared by Craig Counterman, Glenn Golden, Rachel Gollub, Mark Norton, Charles Severance, Lance Speelmon.

[8] A list of Sakai tools is available at https://confluence.sakaiproject.org/pages/viewpage.action?pageId=37290351

[9] Severance, C (2005a) Sakai Technical Update and Futures. August 2, 2005. URL: http://www.dr-chuck.com/talks.php?id=54 [2005_08_02_psu_v01.ppt]

[10] https://confluence.sakaiproject.org/display/MYSAK/Widgets+Home

[11] http://mfeldstein.com/mashing-up-the-lms-the-google-way/

[12] Sakai 2.6 and 3.0: Statement of Ambition, https://confluence.sakaiproject.org/display/REL/Sakai+2.6+and+3.0

[13] Sakai 3 Proposal. A proposal for a next generation Sakai. Note that the proposal is written and distributed by Korcuska, but instead of his name, the title page figures he Sakai Foundation.

[14] https://confluence.sakaiproject.org/display/SAKDEV/Authoring+Contexts

[15] https://confluence.sakaiproject.org/display/SAKDEV/Authoring+Model+Descriptions

[16] https://confluence.sakaiproject.org/display/SAKDEV/Authoring+Taxonomy

[17] https://confluence.sakaiproject.org/display/SAKDEV/Authoring+Summit+-+UI+and+UX+issues

[18] https://confluence.sakaiproject.org/display/SAKDEV/Three+Authoring+Levels

[19] https://confluence.sakaiproject.org/display/SAKDEV/Document+Structure

[20] https://confluence.sakaiproject.org/display/SAKDEV/Authoring+Summit+-+UI+and+UX+issues https://confluence.sakaiproject.org/display/SAKDEV/Three+Authoring+Levels

[21] https://confluence.sakaiproject.org/display/SAKDEV/Authoring+Summit+-+UI+and+UX+issues https://confluence.sakaiproject.org/display/SAKDEV/Three+Authoring+Levels

[22] https://confluence.sakaiproject.org/display/SAKDEV/Document+Structure

[23] https://confluence.sakaiproject.org/display/UX/2008/10/06/Design+requirements+summary

[24] Confluence page, Round 1 design mockups https://confluence.sakaiproject.org/display/UX/2008/10/11/Round+1+design+mock-ups

[25] Confluence page: Getting at the heart of course management and site creation, https://confluence.sakaiproject.org/display/UX/2008/10/20/Getting+at+the+heart+of+course+management+and+site+creation

[26] Email thread: Group discussion. http://sakai-project-mail-list-archives.1343168.n2.nabble.com/Group-discussion-tc2153087.html

[27] Groups, Spaces, Users and Comments https://confluence.sakaiproject.org/pages/viewpage.action?pageId=1114222

[28] Groups, Spaces, Users and Comments https://confluence.sakaiproject.org/pages/viewpage.action?pageId=1114222

[29] Groups, Spaces, Users and Comments https://confluence.sakaiproject.org/pages/viewpage.action?pageId=1114222

[30] Groups, Spaces, Users and Comments
https://confluence.sakaiproject.org/pages/viewpage.action?pageId=1114222
[31] dsafsa
[32] Michael Korcuska accounted for the work being done in 3akai to the Sakai Board of Directors along these lines: "[The project] focuses on a new user experience for key areas of Sakai, especially those areas traditionally associated with project sites. This project has been called 3akai or Sakai 3 and, to date, has focused on […] project site functionality & content authoring." Michael Korcuska: 2009 Sakai Development Process. Sakai Board of Directors Recommendation.
http://mkorcuska.files.wordpress.com/2009/03/sakai-development-process-2009-v05b.pdf
[33] Ray Davis's Communities vs. Sites Confluence page.
https://confluence.sakaiproject.org/display/GROUPS/Communities+vs.+Sites
[34] Clay Fenlason's email comments copied by Korcuska into the page Groups, Spaces, Users and Comments https://confluence.sakaiproject.org/pages/viewpage.action?pageId=1114222
[35] Nathan Pearson's email in the email thread Group discussion http://sakai-project-mail-list-archives.1343168.n2.nabble.com/Group-discussion-tc2153087.html
[36] Scenarios for People, Connections, Profiles,
https://confluence.sakaiproject.org/display/UX/Scenarios+for+People%2C+Connections%2C+Profiles
[37] Nathan Pearson's email in an email exchange reproduced on Confluence
https://confluence.sakaiproject.org/pages/viewpage.action?pageId=79855883
[38] Clay Fenlason's email in the email thread Group discussion http://sakai-project-mail-list-archives.1343168.n2.nabble.com/Group-discussion-tc2153087.html
[39] Ray Davis's Communities vs. Sites Confluence page.
https://confluence.sakaiproject.org/display/GROUPS/Communities+vs.+Sites
[40] Exploratory space for People, Connections, Profile
https://confluence.sakaiproject.org/display/UX/Exploratory+space+for+People%2C+Connections%2C+Profile
[41] Exploratory space for People, Connections, Profile
https://confluence.sakaiproject.org/display/UX/Exploratory+space+for+People%2C+Connections%2C+Profile
[42] Group manager - advanced group creation V2
https://confluence.sakaiproject.org/display/GROUPS/Group+manager+-+advanced+group+creation+V2
[43] Email thread group discussion http://sakai-project-mail-list-archives.1343168.n2.nabble.com/Group-discussion-tc2153087.html
[44] Blog entry Group is not a superclass by Ray Davis.
[45] Daphne Ogle's email comments copied by Korcuska into the page Groups, Spaces, Users and Comments https://confluence.sakaiproject.org/pages/viewpage.action?pageId=1114222
[46] Email thread group discussion http://sakai-project-mail-list-archives.1343168.n2.nabble.com/Group-discussion-tc2153087.html
[47] Ray Davis's Communities vs. Sites Confluence page.
https://confluence.sakaiproject.org/display/GROUPS/Communities+vs.+Sites
[48] Ray Davis's Communities vs. Sites Confluence page.
https://confluence.sakaiproject.org/display/GROUPS/Communities+vs.+Sites
[49] Ray Davis's Communities vs. Sites Confluence page.
https://confluence.sakaiproject.org/display/GROUPS/Communities+vs.+Sites
[50] Clay Fenlason's comment to Ray Davis's Communities vs. Sites Confluence page.
https://confluence.sakaiproject.org/display/GROUPS/Communities+vs.+Sites
[51] https://confluence.sakaiproject.org/display/3AK/BL7+Separate+Sites+from+Groups

[52] http://sakai-project-mail-list-archives.1343168.n2.nabble.com/DG-User-Experience-Proposed-3akai-API-changes-to-improve-site-membership-handling-tt4482086.html#none
[53] Group & workspace relationship https://confluence.sakaiproject.org/pages/viewpage.action?pageId=68162496
[54] https://confluence.sakaiproject.org/display/3AK/%28BL-7%29+Designs+for+sites+distinct+from+groups
[55] ATLAS Network Pilot: Publishing a Formatted File http://www.youtube.com/watch?v=GJcN3TRne5Y&feature=c4-overview&list=UUVEnCEHXGHEIpsCZhqyNn0w
[56] Group creation minispec https://confluence.sakaiproject.org/display/3AK/Group+creation+minispec
[57] From a screencast presentation available at http://www.youtube.com/watch?v=-TgBgsGb5og
[58] Noah Botimer's blog post, Sakai Futures, part 2, http://botimer.net/posts/2010/03/16/sakai-futures-part-2/
[59] Ian Boston in the email discussion Code name for Sakai 3, https://groups.google.com/forum/#!msg/3akai/CvHQd5wF1Rg/X8T3UuVuEAAJ
[60] Nathan Pearson's announcement email, It's a new year and new UX opportunity for Sakai! http://sakai-project-mail-list-archives.1343168.n2.nabble.com/It-s-a-new-year-and-new-UX-opportunity-for-Sakai-td2123739.html
[61] Email thread dev server gets an update. http://sakai-project-mail-list-archives.1343168.n2.nabble.com/DG-User-Experience-Dev-server-gets-an-update-tc2995279.html
[62] Email from the thread Code name for Sakai 3, https://groups.google.com/forum/#!msg/3akai/CvHQd5wF1Rg/X8T3UuVuEAAJ
[63] Email thread: input needed on two new nightly 3akai servers http://sakai-project-mail-list-archives.1343168.n2.nabble.com/DG-User-Experience-input-needed-on-two-new-nightly-3akai-servers-tc4293388.html
[64] See for example Jesse James Garret's framing of web design in the Elements of User Experience (2010), http://www.jjg.net/elements/pdf/elements.pdf and Chapter 5 in the Microsoft online publication, SOA in the Real World. http://msdn.microsoft.com/en-us/library/bb833026.aspx#_Introducing_a_Framework
[65] http://www.jjg.net/elements/pdf/elements.pdf
[66] https://confluence.sakaiproject.org/display/3AK/BL7+Separate+Sites+from+Groups
[67] Confluence page by Keli Amann, Sakai 3.0 capabilities for learning activities, Version of Sep 9, 2009.https://confluence.sakaiproject.org/pages/viewpage.action?pageId=65867799
[68] Confluence page Contextual Inquiry Guides, with downloadable interview protocols for the different roles interviewed. https://confluence.sakaiproject.org/display/UX/Contextual+Inquiry+Guides
[69] Interview protocol document for instructors, entitled CONTEXTUAL INQUIRY SCRIPT – INSTRUCTOR. available for download from the Confluence page https://confluence.sakaiproject.org/display/UX/Contextual+Inquiry+Guides
[70] User and Domain Analysis Part 1 Document, available from the Confluence page https://confluence.sakaiproject.org/display/UX/User+and+Domain+Analysis+Part+1
[71] User and Domain Analysis Part 1 Document, available from the Confluence page https://confluence.sakaiproject.org/display/UX/User+and+Domain+Analysis+Part+1
[72] David Goodrum's guest blog post, Sakai Learning Capabilities Brainstorming, http://mfeldstein.com/sakai-learning-capabilities-brainstorming/
[73] Videocast Josh Baron on Design Lenses, http://vimeo.com/13245171
[74] A Community Process for Requirements Gathering https://confluence.sakaiproject.org/display/USER/A+Community+Process+for+Requirements+Gathering

[75] https://confluence.sakaiproject.org/download/attachments/63766987/Sakai%203%20course%20capabilities%20rev.xlsx?api=v2

[76] Presentation of the Design Lenses diagram on the Confluence page Sakai Learning Capabilities v 1.0, https://confluence.sakaiproject.org/display/PED/Sakai+Learning+Capabilities+v+1.0
The quote also appears as a part of the next citation.

[77] The description of the Design Lenses document on the page of the Sakai Foundation. http://www.sakaiproject.org/design-lenses-group

[78] Confluence page Sakai Learning Capabilities v 1.0, https://confluence.sakaiproject.org/display/PED/Sakai+Learning+Capabilities+v+1.0

[79] See for example the Design Lenses document on the page of the Sakai Foundation. http://www.sakaiproject.org/design-lenses-group, and the presentation at the 2011 Sakai Conference, https://confluence.sakaiproject.org/display/CONF2011/2011-06-15+Sakai+Learning+Capabilities+Design+Lenses+in+Action?src=search

[80] T&L Learning Capabilities Design Lenses Group https://confluence.sakaiproject.org/pages/viewpage.action?pageId=69279066

[81] Lynn E. Wards email summary of an online group call http://collab.sakaiproject.org/pipermail/tl-lenses/2010-August/000003.html

[82] Ian Boston in the email discussion Q1 Server Post Mortem, https://groups.google.com/forum/#!topic/sakai-kernel/BHj6PXZQGgQ

[83] The University of Michigan, Indiana University, Charles Sturt University, the University of Berkeley, and Stanford University stopped their financial contribution to the managed project. Georgia Tech and the University of Cambridge remained, alongside two recently joining non-university partners, rSmart, which had been a Sakai partner and provider since the beginnings of the Sakai Foundation, and the American Academy of Religion.

[84] See Phil Hall's summary of the analyses in a guest blog post entitled Now UC Berkeley and Charles Sturt University Leave Sakai OAE, http://mfeldstein.com/now-uc-berkeley-and-charles-sturt-university-leave-sakai-oae/

[85] David Ackerman's email in the discussion Sakai OAE http://collab.sakaiproject.org/pipermail/openforum/2012-September/000221.html

[86] Nico Matthijs in the email in the discussion Sakai OAE Update, http://collab.sakaiproject.org/pipermail/openforum/2012-October/000256.html

[87] James Farmer's email in the discussion Sakai OAE, http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/IMM/I120907F.pdf

[88] Ian Boston in the email discussion Performance, https://groups.google.com/forum/#!topic/sakai-kernel/xseM17BhsQA

[89] Ian Boston in the email discussion Q1 Server Post Mortem, https://groups.google.com/forum/#!topic/sakai-kernel/BHj6PXZQGgQ

[90] Ian Boston in the email Update on prototypes to solve performance problems, https://groups.google.com/forum/#!topic/sakai-kernel/jednKlXnmWI

[91] Ian Boston's Presentation given at a BOF at Sakai 2011 Conference in LA, http://www.slideshare.net/ianeboston/sparse-content-map-storage-system

[92] Ian Boston in the email discussion Performance, https://groups.google.com/forum/#!topic/sakai-kernel/xseM17BhsQA

[93] Zach Thomas in the email discussion Next steps https://groups.google.com/forum/#!topic/sakai-kernel/FXgGHhunFC0

[94] Ian Boston in the email discussion Q1 Server Post Mortem, https://groups.google.com/forum/#!topic/sakai-kernel/BHj6PXZQGgQ
[95] Ian Boston in the email discussion Q1 Server Post Mortem, https://groups.google.com/forum/#!topic/sakai-kernel/BHj6PXZQGgQ
[96] Ian Boston in the email discussion Complete Rewite or not? https://groups.google.com/forum/#!topic/sakai-kernel/gSMswDyj7G4
[97] Ian Boston in the email discussion Complete Rewite or not? https://groups.google.com/forum/#!topic/sakai-kernel/gSMswDyj7G4
[98] James Renfro in the email discussion Complete Rewite or not? https://groups.google.com/forum/#!topic/sakai-kernel/gSMswDyj7G4
[99] Michael Korcuska in the email discussion The Content Store, and Relational Index, https://groups.google.com/forum/#!topic/sakai-kernel/niSL0Q4UhJU
[100] Ian Boston in the email discussion The Content Store, and Relational Index, https://groups.google.com/forum/#!topic/sakai-kernel/niSL0Q4UhJU
[101] Carl Hall in the e-mail discussion Chat Log Posted, https://groups.google.com/forum/#!topic/sakai-kernel/9JFafsFcs7E
[102] John Norman in the email discussion Where things stand, https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[103] Ian Boston in the email Update on prototypes to solve performance problems, https://groups.google.com/forum/#!topic/sakai-kernel/jednKlXnmWI
[104] Ian Boston in the email Update on prototypes to solve performance problems, https://groups.google.com/forum/#!topic/sakai-kernel/jednKlXnmWI
[105] Ian Boston in the email Sparse Content User Manager, https://groups.google.com/forum/#!topic/sakai-kernel/sWVM_1lqSjg
[106] Ian Boston in the email Replacement search facilities, https://groups.google.com/forum/#!topic/sakai-kernel/Tgx_MW9whjM
[107] Ian Boston in the email Merge Sparse into master? https://groups.google.com/forum/#!topic/sakai-kernel/VmEqlYepML0
[108] OAE technical infrastructure, application & operations, Report by OmniTI, http://ianboston.files.wordpress.com/2012/07/omniti-sakai-oae-report-20120223.pdf
[109] OAE Pilot Performance Issues, https://confluence.sakaiproject.org/display/3AK/OAE+Pilot+Performance+Issues
[110] OAE technical infrastructure, application & operations, Report by OmniTI, http://ianboston.files.wordpress.com/2012/07/omniti-sakai-oae-report-20120223.pdf
[111] Michael Korcuska in the email discussion Where things stand, https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[112] Clay Fenlason in the email discussion Where things stand, https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[113] Michael Korcuska in the email discussion Where things stand, https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[114] Clay Fenlason in the email discussion Where things stand, https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[115] Ian Boston in the email discussion Space
[116] John Norman in the email discussion Space
[117] Ian Boston in the email discussion Space
[118] Ian Boston in the email discussion Space
[119] John Norman in the email discussion Space

[120] Ray Davis in the email discussion Space
[121] Ray Davis in the Confluence page From Sling to Sakai 3 Groups
https://confluence.sakaiproject.org/display/GROUPS/From+Sling+to+Sakai+3+Groups
[122] Ian Boston's email in the email discussion UX review process,
https://groups.google.com/forum/#!topic/sakai-kernel/EjkkrOYGQ7I
[123] Ian Boston's email in the email discussion UX review process,
https://groups.google.com/forum/#!topic/sakai-kernel/EjkkrOYGQ7I
[124] Groups UX Team meeting 8-13-09,
https://confluence.sakaiproject.org/display/GROUPS/Groups+UX+Team+meeting+8-13-09
[125] Daphne Ogle's introduction to the Project Glossary in Confluence
http://confluence.sakaiproject.org/confluence/display/SAKDEV/Project+Glossary
[126] Jun 22, 2009; 7:35pm Daphne Ogle's email Group project use cases and glossary, http://sakai-project-mail-list-archives.1343168.n2.nabble.com/DG-User-Experience-Group-project-use-cases-and-glossary-tc3139346.html
[127] Oliver Heyer, Ray Davis and Keli Amann
[128] http://spreadsheets.google.com/ccc?key=rXb1dLcREdpWX5hUm2INCnQ&hl=en
[129] Joanna Proulx, Keli Amann, Barbra Mack and Eli Cochran
[130] "Officialish" groups in Courses, "Officialish" groups on campus, Campus Affiliation, Student Project Groups, Outside Community, Common interest /affiliation
[131] http://spreadsheets.google.com/ccc?key=rXb1dLcREdpWX5hUm2INCnQ&hl=en
[132] Daphne Ogle's introduction to the transcript in the Confluence page Mental Models – Groups
https://confluence.sakaiproject.org/display/GROUPS/Mental+Models+-+Groups
[133]
https://confluence.sakaiproject.org/display/GROUPS/One+Person%27s+Mental+Model+of+Workspaces+and+Groups
[134] Ray Davis' comment on the Confluence page Mental Models – Groups
https://confluence.sakaiproject.org/display/GROUPS/Mental+Models+-+Groups
[135] Ray Davis' comment on the Confluence page Mental Models – Groups
https://confluence.sakaiproject.org/display/GROUPS/Mental+Models+-+Groups
[136] Ray Davis' comment on Keli Amann's presentation, One Person's Mental Model of Workspaces and Groups
https://confluence.sakaiproject.org/display/GROUPS/One+Person%27s+Mental+Model+of+Workspaces+and+Groups
[137] Ray Davis' comment on Keli Amann's presentation, One Person's Mental Model of Workspaces and Groups
https://confluence.sakaiproject.org/display/GROUPS/One+Person%27s+Mental+Model+of+Workspaces+and+Groups
[138] Ray Davis' comment on Keli Amann's presentation, One Person's Mental Model of Workspaces and Groups
https://confluence.sakaiproject.org/display/GROUPS/One+Person%27s+Mental+Model+of+Workspaces+and+Groups
[139] Ray Davis' comment on Keli Amann's presentation, One Person's Mental Model of Workspaces and Groups
https://confluence.sakaiproject.org/display/GROUPS/One+Person%27s+Mental+Model+of+Workspaces+and+Groups
[140] Ray Davis' comment on Keli Amann's presentation, One Person's Mental Model of Workspaces and Groups

https://confluence.sakaiproject.org/display/GROUPS/One+Person%27s+Mental+Model+of+Workspaces+and+Groups
[141] Ray Davis' comment on Keli Amann's presentation, One Person's Mental Model of Workspaces and Groups
https://confluence.sakaiproject.org/display/GROUPS/One+Person%27s+Mental+Model+of+Workspaces+and+Groups
[142] Ray Davis' comment on Keli Amann's presentation, One Person's Mental Model of Workspaces and Groups
https://confluence.sakaiproject.org/display/GROUPS/One+Person%27s+Mental+Model+of+Workspaces+and+Groups
[143] Ray Davis' comment on Keli Amann's presentation, One Person's Mental Model of Workspaces and Groups
https://confluence.sakaiproject.org/display/GROUPS/One+Person%27s+Mental+Model+of+Workspaces+and+Groups
[144] Clay Fenlason's comment on the Confluence page Round 1 design mock-ups
https://confluence.sakaiproject.org/display/UX/2008/10/11/Round+1+design+mock-ups
[145] John Norman, comment on the Confluence page Mental Models – Groups
https://confluence.sakaiproject.org/display/GROUPS/Mental+Models+-+Groups
[146] The results were published on the Confluence page Federated Authorization Use Cases, which is the source for all the examples cited in this section.
https://confluence.sakaiproject.org/display/GROUPS/Federated+Authorization+Use+Cases
[147] https://confluence.sakaiproject.org/display/GROUPS/SAKAI09+Groups-Roles+Notes
[148] Kristol Hancock's account on the Confluence page for benchmarking systems
https://confluence.sakaiproject.org/display/GROUPS/Benchmarking+-+Comparing+how+other+systems+deal+with+groups
[149] Eli Cochran's account on the Confluence page for benchmarking systems
https://confluence.sakaiproject.org/display/GROUPS/Benchmarking+-+Comparing+how+other+systems+deal+with+groups
[150] Ray Davis' account on the Confluence page for benchmarking systems
https://confluence.sakaiproject.org/display/GROUPS/Benchmarking+-+Comparing+how+other+systems+deal+with+groups
[151] Daphne Ogle's introduction to the Confluence page Context Scenarios (Group creation & management)
https://confluence.sakaiproject.org/pages/viewpage.action?pageId=16220197
[152] Lynn Ward's scenario on the Confluence page Context Scenarios (Group creation & management)
https://confluence.sakaiproject.org/pages/viewpage.action?pageId=16220197
[153] Authoring Summit – Use Cases Confluence page,
https://confluence.sakaiproject.org/display/SAKDEV/Authoring+Summit+-+Use+Cases
[154] Confluence page JCR rationale and implications,
https://confluence.sakaiproject.org/display/CHS/JCR+rationale+and+implications
[155] Confluence page JCR rationale and implications,
https://confluence.sakaiproject.org/display/CHS/JCR+rationale+and+implications
[156] John Norman in the email discussion Where things stand,
https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[157] Ian Boston in the email discussion Complete Rewite or not?
https://groups.google.com/forum/#!topic/sakai-kernel/gSMswDyj7G4
[158] Michael Korcuska in the email discussion The Content Store, and Relational Index,
https://groups.google.com/forum/#!topic/sakai-kernel/niSL0Q4UhJU

[159] Ian Boston in the email discussion Where things stand, https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[160] Thomas Amsler in the e-mail discussion Chat Log Posted, https://groups.google.com/forum/#!topic/sakai-kernel/9JFafsFcs7E
[161] Ian Boston in the e-mail discussion Chat Log Posted, https://groups.google.com/forum/#!topic/sakai-kernel/9JFafsFcs7E
[162] John Norman in the email discussion Where things stand, https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[163] Paul Bristow in the email discussion Where things stand. https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[164] Thomas Amsler in the email discussion Where things stand. https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[165] Jon Gorrono in the email discussion Where things stand. https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[166] Ian Boston in the email discussion Sakai and Sling, https://groups.google.com/forum/#!topic/sakai-kernel/FM7TUuHu5WU
[167] Thomas Amsler in the email discussion Where things stand. https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[168] Paul Bristow's email in the thread Where things stand https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[169] Chris J. Holdorphs's email in the thread Where things stand https://groups.google.com/forum/#!topic/sakai-kernel/B7uA-vpoReM
[170] Mark J. Norton's exploratory suggestion for Sakai Repository API Design https://confluence.sakaiproject.org/display/DOC/Repository+Service+Documentation
[171] Ian Boston's comment on the blog post by Phil Hill, Now UC Berkeley and Charles Sturt University Leave Sakai OAE, http://mfeldstein.com/now-uc-berkeley-and-charles-sturt-university-leave-sakai-oae/ http://mfeldstein.com/now-uc-berkeley-and-charles-sturt-university-leave-sakai-oae/
[172] Michael Feldstein blog post on his blog, the e-Literate, entitled Sakai 3: The Benefits of 'Everything is Content.' http://mfeldstein.com/sakai-3-the-benefits-of-everything-is-content/
[173] Mark J. Norton outlined a suggested design for a Sakai Repository based on JSR-170 in March of 2005, see the related email http://thread.gmane.org/gmane.comp.cms.sakai.devel/2779/focus=2791
[174] Mark J. Norton's exploratory suggestion for Sakai Repository API Design https://confluence.sakaiproject.org/display/DOC/Repository+Service+Documentation
[175] Oliver Heyer and Ray Davis on Confluence page Draft -- The Transition to File Management, https://confluence.sakaiproject.org/display/CHS/Draft+--+The+Transition+to+File+Management
[176] Prioritization for post v1 work on One Big Thing Use Cases https://confluence.sakaiproject.org/display/3AK/Prioritization+for+post+v1+work+on+One+Big+Thing+Use+Cases
[177] http://blog.tfd.co.uk/2006/07/27/contenthosting-jsr-170/
[178] JSR-170 and Content Hosting & Repository Integration, Ian Boston's presentation at the 8th Sakai Conference in December 2007, https://confluence.sakaiproject.org/download/attachments/40108043/ContentHosting20071129.pdf?version=1&modificationDate=1197572353000&api=v2
[179] JSR-170 and Content Hosting & Repository Integration, Ian Boston's presentation at the 8th Sakai Conference in December 2007, https://confluence.sakaiproject.org/download/attachments/40108043/ContentHosting20071129.pdf?version=1&modificationDate=1197572353000&api=v2

[180] Clay Fenlason's account on the Confluence page Cambridge Get-Together https://confluence.sakaiproject.org/display/RES/Cambridge+Get-Together
[181] Michael Feldstein's blog post Mashing Up the LMS the Google Way http://mfeldstein.com/mashing-up-the-lms-the-google-way/
[182] John Norman on Confluence page JCR rationale and implications, https://confluence.sakaiproject.org/display/CHS/JCR+rationale+and+implications
[183] Clay Fenlason in the email discussion K2 Incubation proposal, https://groups.google.com/forum/#!topic/sakai-kernel/8J5cyS_hvKs
[184] Laura James in the email K2 scope and "3.0", https://groups.google.com/forum/#!topic/sakai-kernel/CMbVkWOCI7E
[185] David Adams, Director of Server and Network Operations at TLOS, Virginia Tech, 2012-09-07
[186] http://www.sakaiproject.org/organization-list, accessed on the 22nd of May, 2013.